VISUALISATION DE DONNÉES SUR SEABORN

Thomas Caro

SOMMAIRE

MISE EN FORME DES DONNÉES

TYPE DE DONNÉES

Dataframe Pandas/liste Numpy/liste python

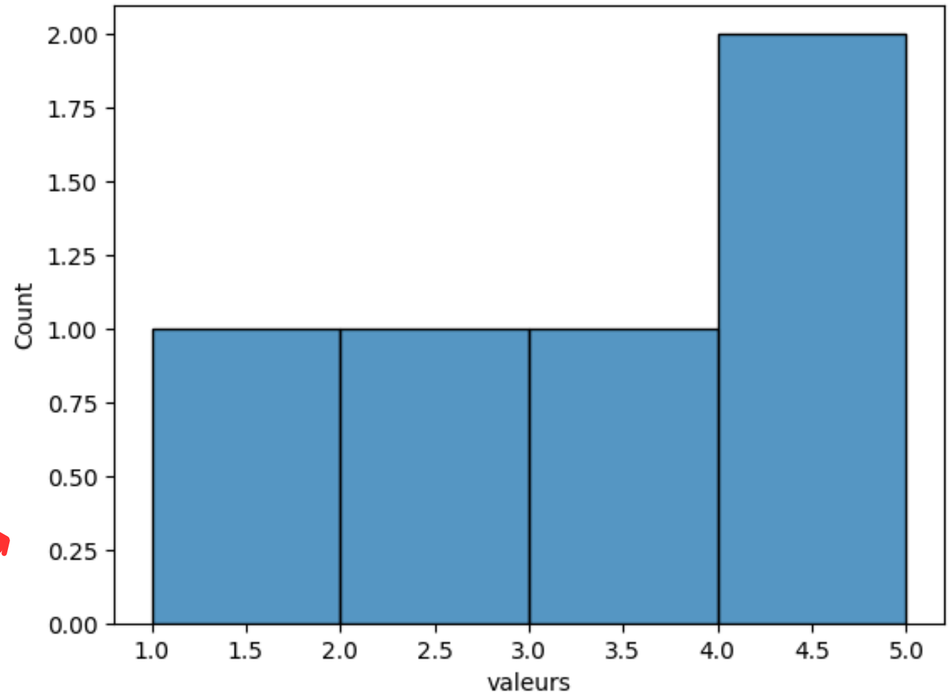
Les 3 fontionnent et peuvent donner le même résultat.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data = np.array([1, 2, 3, 4, 5])
sns.histplot(data)
plt.show()

data = [1, 2, 3, 4, 5]
sns.histplot(data)
plt.show()

df = pd.DataFrame({"valeurs": [1, 2, 3, 4, 5]})
sns.histplot(data=df, x="valeurs")
plt.show()
```



Mais le format à privilégier est le **dataframe panda**, c'est celui qui offrira le plus d'options le plus facilement.

ISDM
INSTITUT DE SCIENCE DES DONNÉES
MONTPELLIER

FORMATS DE DONNÉES POUR SEABORN

DataFrame : wide-form data

Une variable par axe, et la troisième dans les cases du tableau.

Var1	Valeur1	Valeur2	Valeur3
Var2			
Valeur1	**Var3**Val11	**Var3**Val12	**Var3**Val13
Valeur2	**Var3**Val21	**Var3**Val22	**Var3**Val23
Valeur3	**Var3**Val31	**Var3**Val32	**Var3**Val33

DataFrame : long-form data

- Chaque colonne est une variable
- Chaque ligne est une observation

	Var1	Var2	Var3
Obs1			
Obs2			
Obs3			

LONG-FORM DATA

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult male	deck	embark town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

Chaque ligne est désignée par son index. Ces index ne sont pas considérés comme une colonne.

Les entêtes des colonnes, de manière similaire, ne sont là que pour décrire les colonnes mais ne sont pas une ligne à part entière.

Enfin, ici on retrouve effectivement les données/observations.

L'INDEX

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult male	deck	embark town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

Par défault, les index sont des entiers commençant à 0. Mais ils peuvent être changé à la main pour prendre d'autres valeurs.

Pour accéder à une ligne avec l'index : tableau.iloc[index]

Pour accéder à une portion* de ligne : tableau.iloc[index_1 : index_2]

Pour accéder à plusieurs lignes précises : tableau.iloc[[index_1, index_2, index_3, …]]

*Attention, la borne supérieure n'est jamais incluse donc tableau.iloc[5 :18] ne ressortira que les lignes avec un index de 5 à 17. De plus, si vous voulez commencer à 0, vous pouvez laisser vide [:18] et pareil pour la fin [25:].

LES ENTÊTES DE COLONNES

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult male	deck	embark town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

Pour accéder à une colonne : tableau[“Nom de la colonne”]

Pour accéder à certains index d’une colonne : tableau[“nom colonne”].iloc[index]

Dans la documentation de Seaborn, les entêtes des colonnes sont référés par “variables”, donc “names of variables” correspond simplement au nom/entête de la colonne.

Lorsqu’une ligne/colonne est extraite, elle devient une Series, un autre format de données de Seaborn.

DONNÉES MANQUANTES

Solutions

Tout d'abord vérifier si des données sont manquantes à l'aide de pandas :

```
data=sns.load_dataset("penguins")
print(data.isnull())#sur le tableau entier
print(data.isnull().any())#sur chaque colonne
```

Puis si l'on ne veut pas considérer les observations avec valeurs nulles pour une variable par exemple :

```
data.dropna(subset=["body_mass_g"])
```

Ici on regarde seulement la variable body_mass_g, et on retire les observations qui n'ont pas de valeur pour celle-ci.

DONNÉES QUANTITATIVES

LES DONNÉES QUANTITATIVES

Que sont les données numériques?

Les données **quantitatives**, ou **numériques**, comme leur nom l'indique, sont constituées des données **mesurables**, donc des **nombres**.

FONCTIONS SEABORN

Il y a différents types de plot pour les données quantitatives sur seaborn :

- *relplot(scatterplot et lineplot)*
- *displot(histplot, kdeplot et ecdfplot)*
- *boxplot*
- *violinplot*
- *regplot et lmplot*

RELPLOT

Définition

seaborn.relplot(data=None, *, x=None, y=None, hue=None, size=None, style=None, units=None, weights=None, row=None, col=None, col_wrap=None, row_order=None, col_order=None, palette=None, hue_order=None, hue_norm=None, sizes=None, size_order=None, size_norm=None, markers=None, dashes=None, style_order=None, legend='auto', kind='scatter', height=5, aspect=1, facet_kws=None, **kwargs)

RELPLOT

Définition

seaborn.relplot(data=None, *, x=None, y=None, hue=None, size=None, style=None, units=None, weights=None, row=None, col=None, col_wrap=None, row_order=None, col_order=None, palette=None, hue_order=None, hue_norm=None, sizes=None, size_order=None, size_norm=None, markers=None, dashes=None, style_order=None, legend='auto', kind='scatter', height=5, aspect=1, facet_kws=None, **kwargs)

Il y évidemment une documentation accessible sur internet, donc on ne va passer que sur les éléments les plus essentiels pour pouvoir afficher ce dont on a besoin le plus vite possible, à savoir :

RELPLOT

Définition

seaborn.relplot(**data=None**, *, **x=None**, **y=None**, **hue=None**, **size=None**, **style=None**, units=None, weights=None, **row=None**, **col=None**, col_wrap=None, row_order=None, col_order=None, palette=None, hue_order=None, hue_norm=None, sizes=None, size_order=None, size_norm=None, markers=None, dashes=None, style_order=None, legend='auto', **kind='scatter'**, height=5, aspect=1, facet_kws=None, **kwargs)

RELPLOT

Paramètres

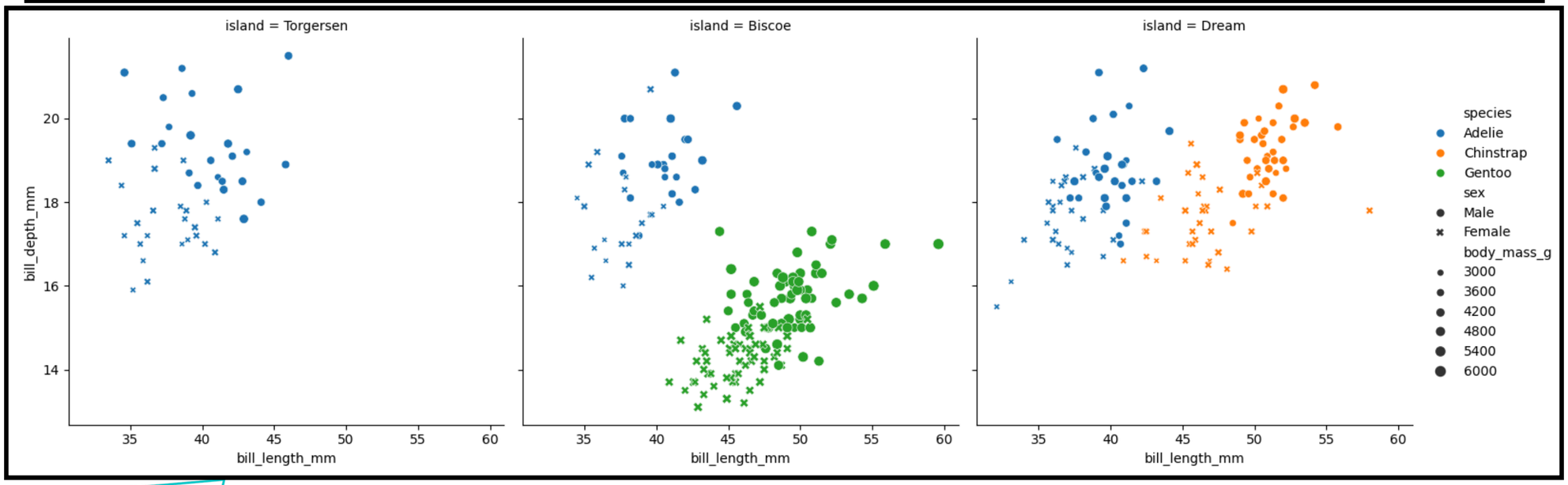
Nom du paramètre	Explications	Ce qui lui faut comme format	Exemple
data	Il faut donner au paramètre *data* le tableau entier que vous traitez	DataFrame, Series, dict, array, or list of arrays	data=tableau
x	variable du tableau utilisée pour les abscisses	Chaine de caractères correspondant à une variable du tableau	x="poids"
y	variable du tableau utilisée pour les ordonnées	Chaine de caractères correspondant à une variable du tableau	y=”taille”
hue	variable du tableau permettant de rajouter une dimension avec de la couleur	Chaine de caractères correspondant à une variable du tableau, catégorielle ou entière	hue=”age”
size	variable du tableau qui controlera la taille des points	Chaine de caractères correspondant à une variable du tableau, numérique	size=”argent”
style	variable du tableau qui controlera le style des points	Chaine de caractères correspondant à une variable du tableau, catégorielle ou entière	style=”sex”
row	variable du tableau qui permettra de créer un tableau de graphiques, ici les lignes	Chaine de caractères correspondant à une variable du tableau, catégorielle ou entière	row=”catégorie”
col	variable du tableau qui permettra de créer un tableau de graphiques, ici les colonnes	Chaine de caractères correspondant à une variable du tableau, catégorielle ou entière	col=”métier”
kind	type de graphique que l’on veut	Chaine de caractères, 2 choix possibles	kind=”scatter” ou kind=”line”

RELPLOT

Exemple : scatter

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

data = sns.load_dataset("penguins")
sns.relplot(data=data,x="bill_length_mm",y="bill_depth_mm",hue="species",style="sex",size="body_mass_g",col="island")
plt.show()
```



kind par défaut vaut “scatter”

RELPLOT

Exemple : scatter

On peut rajouter des ellipses sur des relplot scatter, pour dessiner dessus il faut récupérer l'*ax* :

```
df = sns.load_dataset("penguins")
df = df[["species", "bill_length_mm", "body_mass_g"]].dropna()

g = sns.relplot(
    data=df,
    x="bill_length_mm",
    y="body_mass_g",
    hue="species",
    kind="scatter",
    height=5
)
ax = g.ax
```

```
def add_confidence_ellipse(x, y, ax, n_std=2.0, **kwargs):
    cov = np.cov(x, y)
    mean = np.mean(x), np.mean(y)
    eigvals, eigvecs = np.linalg.eigh(cov)
    order = eigvals.argsort()[::-1]
    eigvals, eigvecs = eigvals[order], eigvecs[:, order]
    angle = np.degrees(np.arctan2(*eigvecs[:, 0][::-1]))
    width, height = 2 * n_std * np.sqrt(eigvals)
    ellipse = Ellipse(
        xy=mean,
        width=width,
        height=height,
        angle=angle,
        fill=False,
        **kwargs
    )
    ax.add_patch(ellipse)
```

Fonction de calcul et d'ajout de l'ellipse , *Ellipse* vient de matplotlib.patches

RELPLOT

Exemple : scatter

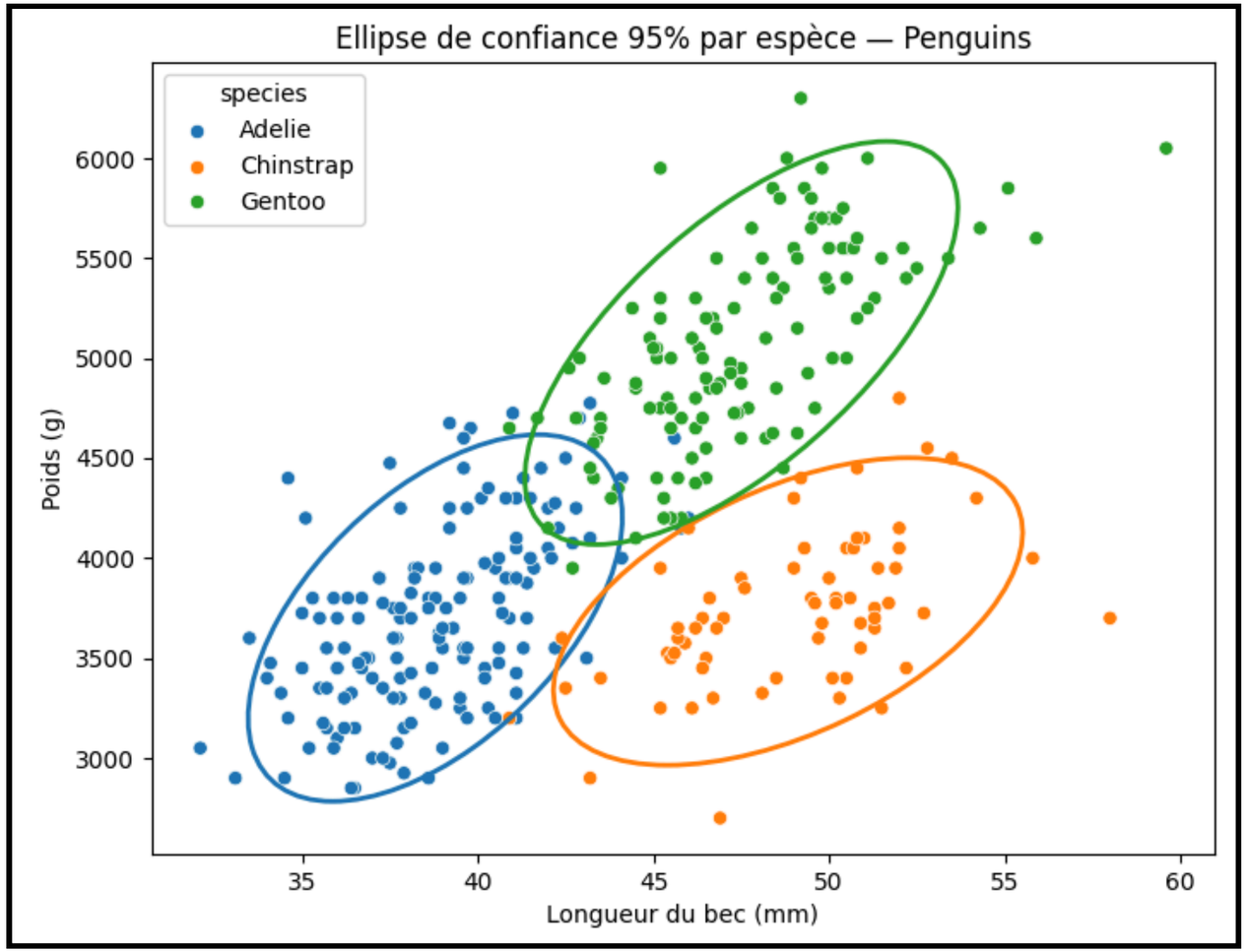
Et puis calculer et afficher l'ellipse pour chaque espèce :

```
# Ajouter une ellipse pour chaque espèce
palette = sns color_palette()  # Même palette que pour hue

for i, species in enumerate(df["species"].unique()):
    subset = df[df["species"] == species]
    # Calculer l'ellipse
    add_confidence_ellipse(
        subset["bill_length_mm"],
        subset["body_mass_g"],
        ax,
        edgecolor=palette[i],
        linewidth=2
    )

ax.set_xlabel("Longueur du bec (mm)")
ax.set_ylabel("Poids (g)")
ax.set_title("Ellipse de confiance 95% par espèce — Penguins")

plt.show()
```

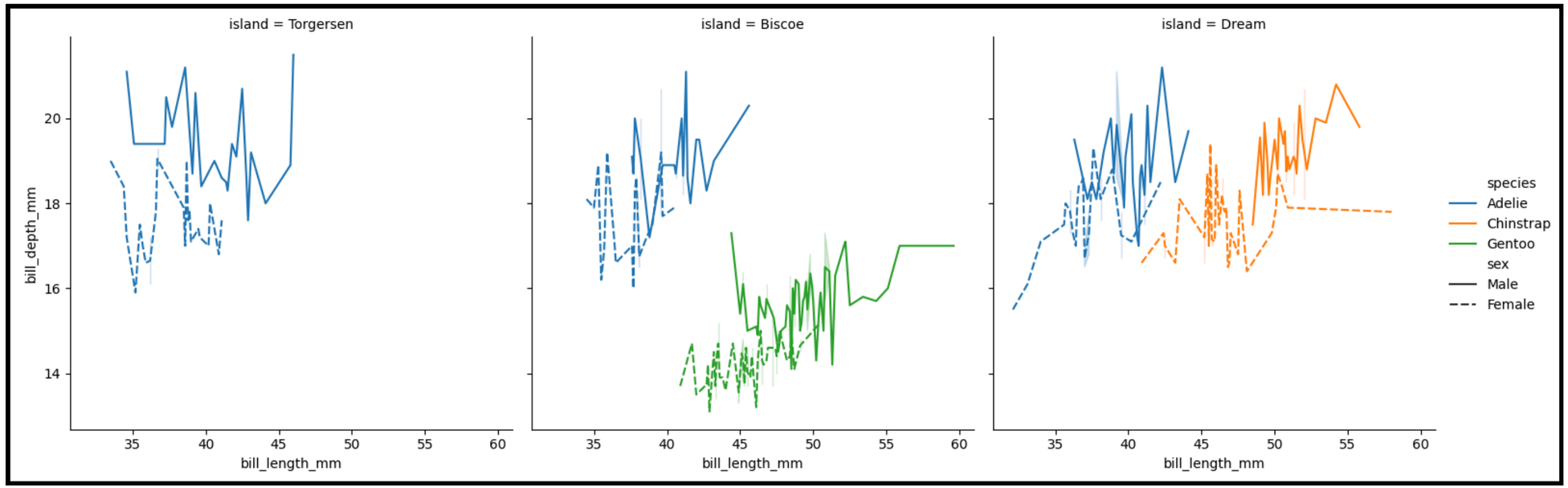


RELPLOT

Exemple : line

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

df = sns.load_dataset("penguins")
sns.relplot(data=df,x="bill_length_mm",y="bill_depth_mm",hue="species",style="sex",col="island",kind="line")
plt.show()
```



Les données ne se prêtent pas bien pour des “lines”,
size n’est pas utilisable avec des lines.

DISPLOT

Définition

seaborn.displot(**data=None**, *, **x=None**, **y=None**, **hue=None**, **row=None**, **col=None**, weights=None, **kind='hist'**, **rug=False**, rug_kws=None, log_scale=None, legend=True, palette=None, hue_order=None, hue_norm=None, color=None, col_wrap=None, row_order=None, col_order=None, height=5, aspect=1, facet_kws=None, **kwargs)

DISPLOT

Paramètres

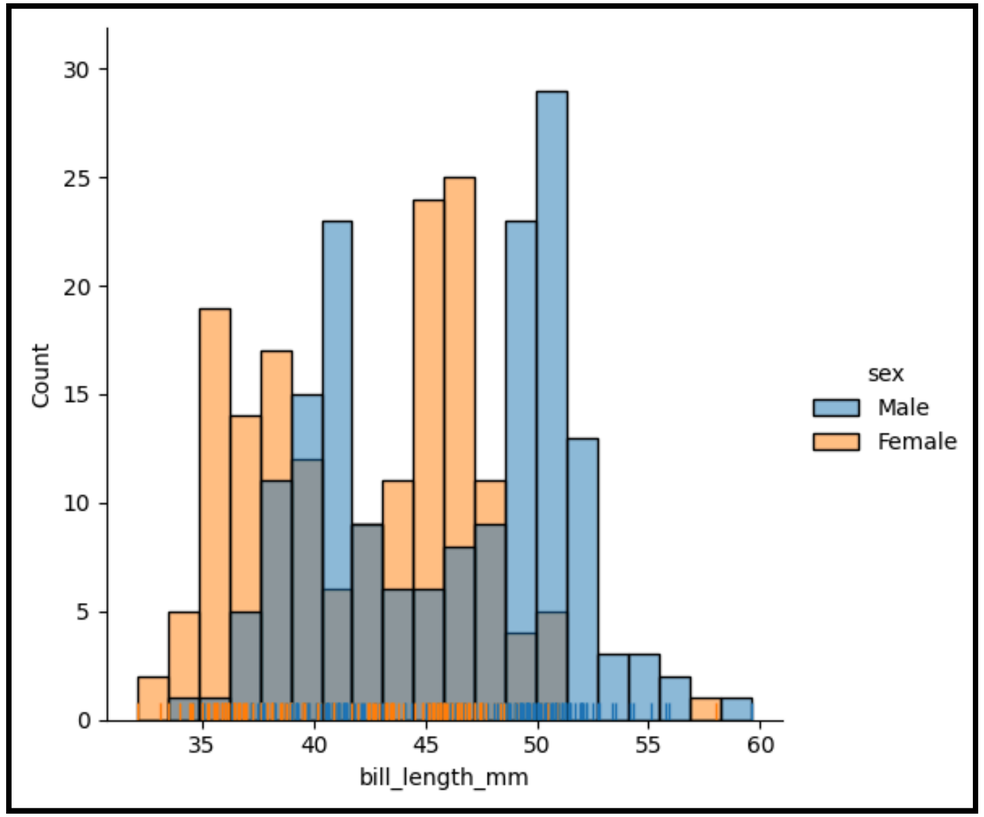
Nom du paramètre	Explications	Ce qui lui faut comme format	Exemple
data	Il faut donner au paramètre *data* le tableau entier que vous traitez	DataFrame, Series, dict, array, or list of arrays	data=tableau
x	variable du tableau utilisée pour les abscisses	Chaine de caractères correspondant à une variable du tableau	x="poids"
y	variable du tableau utilisée pour les ordonnées	Chaine de caractères correspondant à une variable du tableau	y=”taille”
hue	variable du tableau permettant de rajouter une dimension avec de la couleur	Chaine de caractères correspondant à une variable du tableau, catégorielle ou entière	hue=”age”
row	variable du tableau qui permettra de créer un tableau de graphiques, ici les lignes	Chaine de caractères correspondant à une variable du tableau, catégorielle ou entière	row=”catégorie”
col	variable du tableau qui permettra de créer un tableau de graphiques, ici les colonnes	Chaine de caractères correspondant à une variable du tableau, catégorielle ou entière	col=”métier”
kind	type de graphe que l’on veut	Chaine de caractères, 3 choix possibles	kind=”hist”,kind=”kde” ou kind=”ecdf”
rug	permet de voir les observations individuelles sur les axes.	Booléen	rug=True

DISPLOT

Exemple : hist

Mêmes importations que précédemment :

```
data = sns.load_dataset("penguins")
sns.displot(data=data, x="bill_length_mm", rug=True, hue="sex", bins=20)
plt.show()
```



Si l'on ne renseigne pas la donnée à mettre en ordonnée (*y*), l'ordonnée sera le nombre d'occurence, et si l'on ne renseigne pas le *kind* c'est un histogramme par défaut.

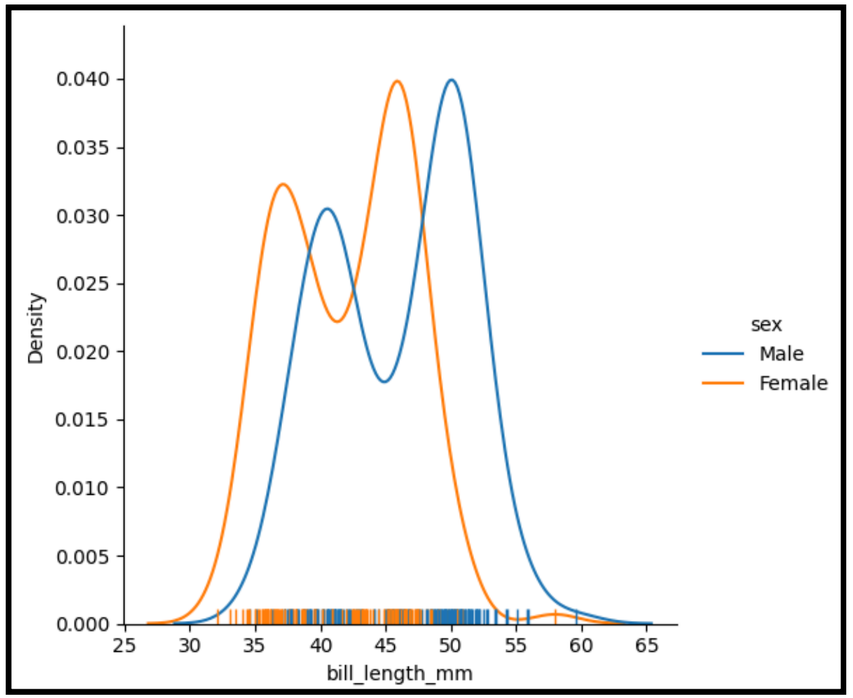
L'argument *bins* controle le nombre de barres.

DISPLOT

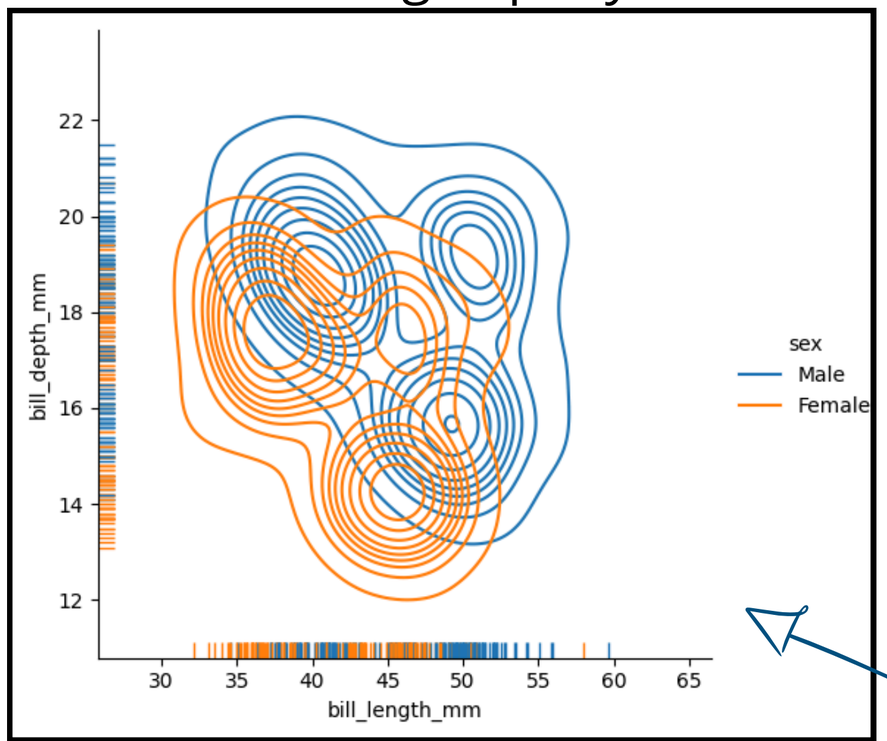
Exemple : kde(kernel density estimate)

Mêmes importations que précédemment :

```
data = sns.load_dataset("penguins")
sns.displot(data=data,x="bill_length_mm", rug=True, hue="sex", kind="kde")
plt.show()
```



Si l'on ne renseigne pas *y* nous avons la densité de probabilité.



Si on renseigne *y* nous avons un kde en 2 dimensions

```
data = sns.load_dataset("penguins")
sns displot(data=data,x="bill_length_mm", y="bill_depth_mm"  rug=True, hue="sex"  kind="kde")
plt.show()
```

DISPLOT

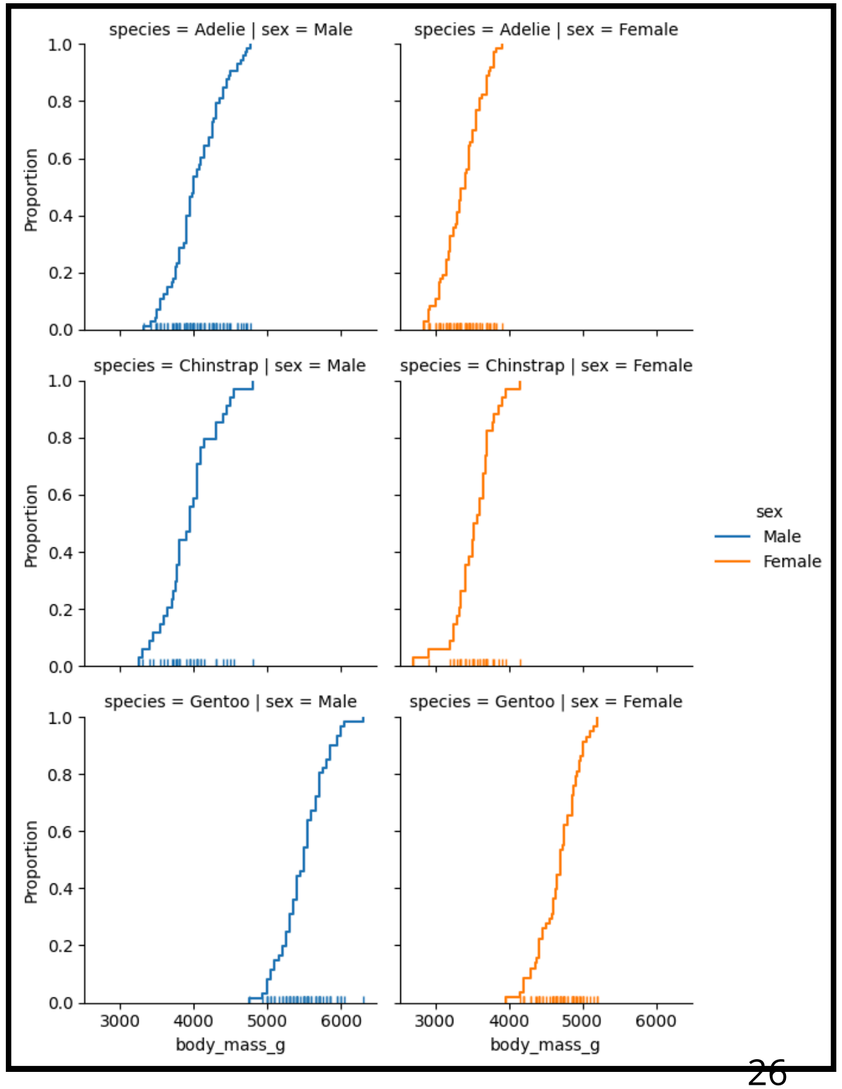
Exemple : ecdf(empirical distribution function)

```
data = sns.load_dataset("penguins")
sns.displot(data=data, x="body_mass_g", rug=True, hue="sex", kind="ecdf", row="species", col="sex",
height=5)
plt.show()
```

L'ecdf est univariationnelle, on ne renseigne donc qu'un *x*.

Les paramètre *col* et *row* permettent de visualiser 2 autres variables.

Le paramètre *height* contrôle la hauteur des graphiques.



BOXPLOT

Définition

seaborn.boxplot(**data=None**, *, **x=None**, **y=None**, **hue=None**, order=None, hue_order=None, orient=None, color=None, palette=None, saturation=0.75, fill=True, **dodge='auto'**, **width=0.8**, **gap=0**, whis=1.5, linecolor='auto', linewidth=None, fliersize=None, hue_norm=None, native_scale=False, log_scale=None, formatter=None, legend='auto', ax=None, **kwargs)

BOXPLOT

Paramètres

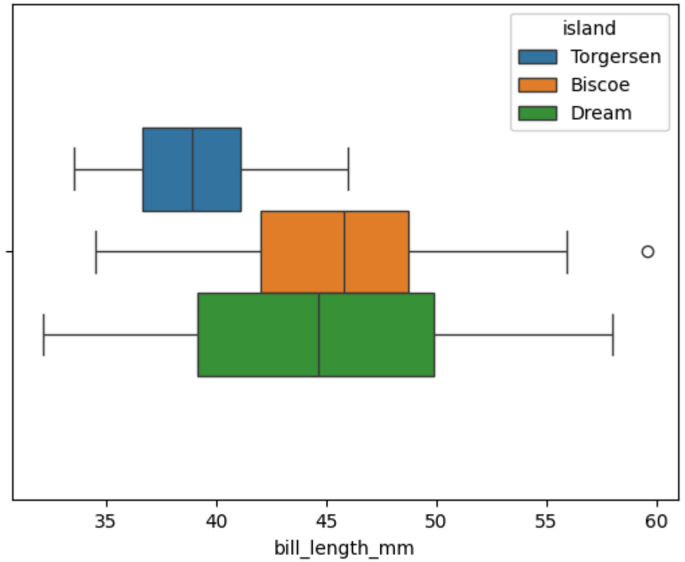
Nom du paramètre	Explications	Ce qui lui faut comme format	Exemple
data	Il faut donner au paramètre *data* le tableau entier que vous traitez	DataFrame, Series, dict, array, or list of arrays	data=tableau
x	variable du tableau utilisée pour les abscisses	Chaine de caractères correspondant à une variable du tableau	x="poids"
y	variable du tableau utilisée pour les ordonnées	Chaine de caractères correspondant à une variable du tableau	y=”taille”
hue	variable du tableau permettant de rajouter une dimension avec de la couleur	Chaine de caractères correspondant à une variable du tableau, catégorielle ou entière	hue=”age”
dodge	variable permettant de choisir si les graphiques peuvent se superposer	Booléen	dodge=False
width	Variable permettant de contrôler les largeurs des boîtes.	Valeur flottante	width=0.5
gap	Variable permettant de contrôler l’écart entre les différentes boîtes “dodgées”	Valeur flottante	gap=0.1

BOXPLOT

Exemples

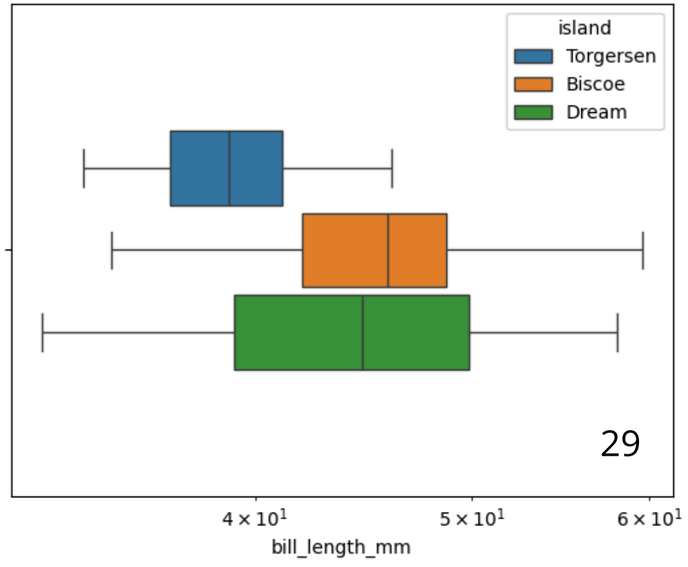
```
data = sns.load_dataset("penguins")
sns boxplot(data=data, x="bill_length_mm", hue="island", dodge=True  width=0.5)
plt show()
```

De base *gap* vaut 0. L'orientation est gérée automatiquement par seaborn, mais si le graphique est bidimensionnel on peut la chosir.




```
data = sns.load_dataset("penguins")
sns.boxplot(data=data, x="bill_length_mm", hue="island", dodge=True, width=0.5, gap=0.1, log_scale=True)
plt.show()
```

log_scale permet de changer l'échelle. Une valeur numérique définit la base. Si le graphique est bidimensionnel 2 valeurs peuvent être données, une pour chaque axe.



VIOLINPLOT

Définition

seaborn.violinplot(**data=None**, *, **x=None**, **y=None**, **hue=None**, order=None, hue_order=None, orient=None, color=None, palette=None, saturation=0.75, fill=True, **inner='box'**, **split=False**, **width=0.8**, **dodge='auto'**, **gap=0**, linewidth=None, linecolor='auto', cut=2, gridsize=100, bw_method='scott', bw_adjust=1, density_norm='area', common_norm=False, hue_norm=None, formatter=None, log_scale=None, native_scale=False, legend='auto', scale=<deprecated>, scale_hue=<deprecated>, bw=<deprecated>, inner_kws=None, ax=None, **kwargs)

VIOLINPLOT

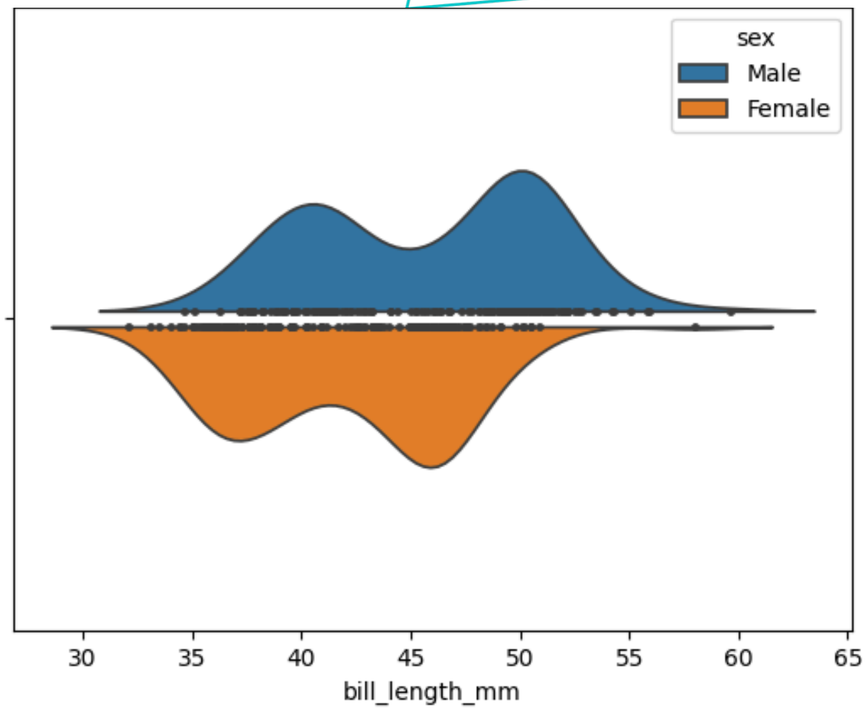
Paramètres

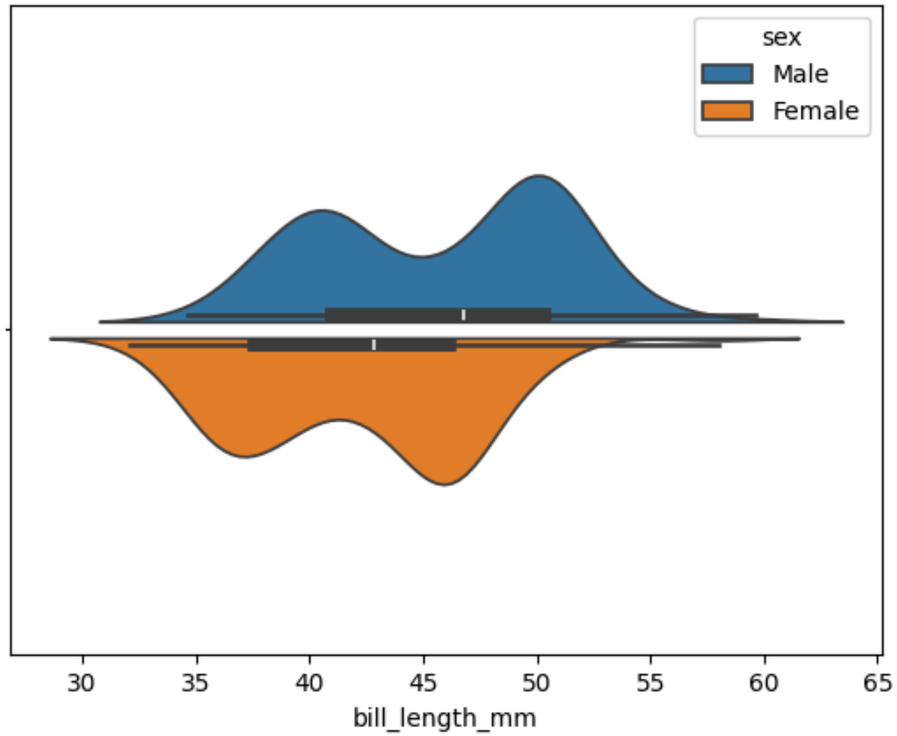
Nom du paramètre	Explications	Ce qui lui faut comme format	Exemple
data	Il faut donner au paramètre *data* le tableau entier que vous traitez	DataFrame, Series, dict, array, or list of arrays	data=tableau
x	variable du tableau utilisée pour les abscisses	Chaine de caractères correspondant à une variable du tableau	x="poids"
y	variable du tableau utilisée pour les ordonnées	Chaine de caractères correspondant à une variable du tableau	y=”taille”
hue	variable du tableau permettant de rajouter une dimension avec de la couleur	Chaine de caractères correspondant à une variable du tableau, catégorielle ou entière	hue=”age”
inner	variable permettant de contrôler la représentation des données dans le violon	Chaine de caractères correspondant à un type de représentation	inner=”box”,inner=”quart”,inner=”point”
split	Variable permettant de choisir si la représentation est symétrique.	Booléen	split=True
width	Variable permettant de contrôler la largeur des violons.	Valeur flottante	width=0.5
dodge	variable permettant de choisir si les graphes peuvent se superposer	Booléen	dodge=False
gap	Variable permettant de contrôler l’écart entre les différentes boîtes “dodgées”	Valeur flottante	gap=0.1

VIOLINPLOT

Exemple

```
data = sns.load_dataset("penguins")
sns.violinplot(data=data, x="bill_length_mm", hue="sex", dodge=True, linewidth=3,
split=True, inner="point")
plt.show()
```






```
data = sns.load_dataset("penguins")
sns.violinplot(data=data, x="bill_length_mm", hue="sex",dodge=True,linewidth=3,
split=True, inner="box")
plt.show()
```

REGPLOT

Définition

seaborn.regplot(**data=None**, *, **x=None**, **y=None**, x_estimator=None, x_bins=None, x_ci='ci', scatter=True, fit_reg=True, **ci=95**, **n_boot=1000**, units=None, **seed=None**, order=1, **logistic=False**, **lowess=False**, **robust=False**, logx=False, x_partial=None, y_partial=None, truncate=True, dropna=True, x_jitter=None, y_jitter=None, label=None, color=None, marker='o', scatter_kws=None, line_kws=None, ax=None)

REGPLOT

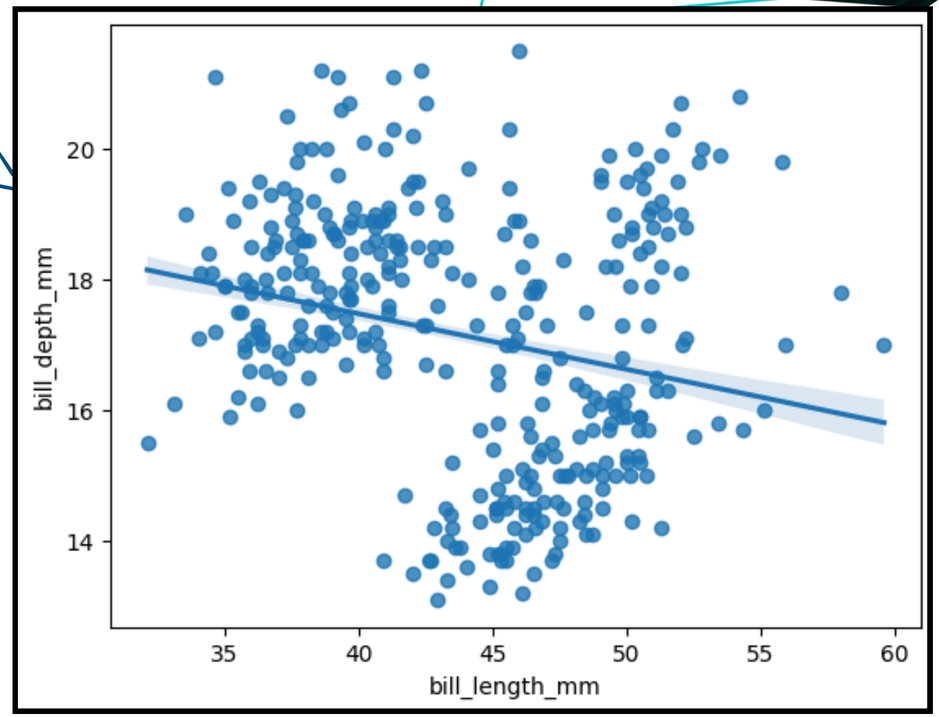
Paramètres

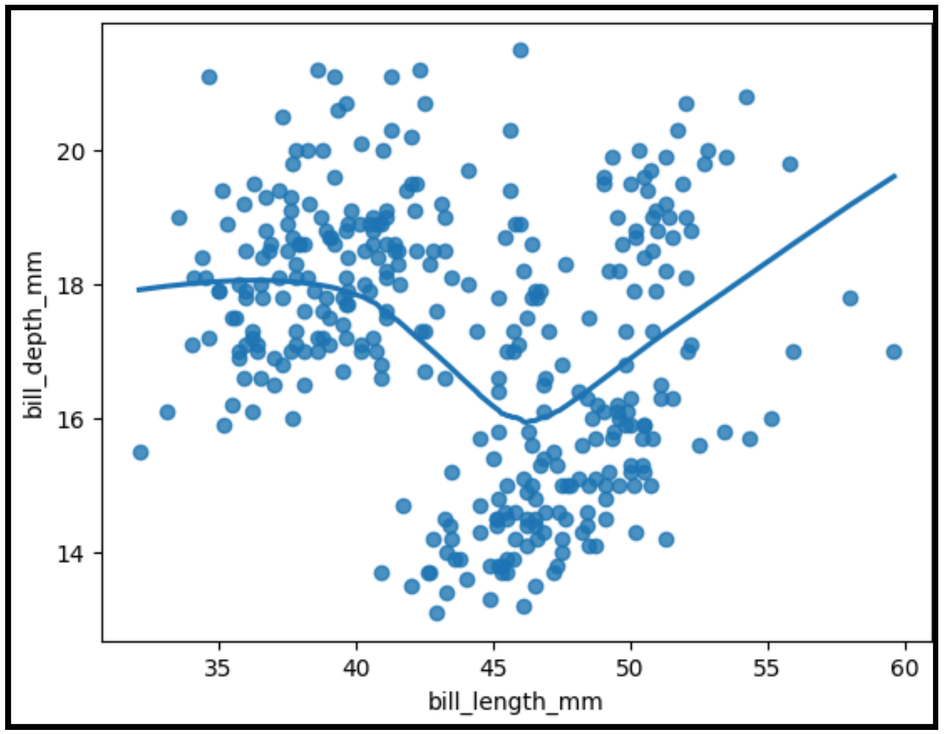
Nom du paramètre	Explications	Ce qui lui faut comme format	Exemple
data	Il faut donner au paramètre *data* le tableau entier que vous traitez	DataFrame, Series, dict, array, or list of arrays	data=tableau
x	variable du tableau utilisée pour les abscisses	Chaine de caractères correspondant à une variable du tableau	x="poids"
y	variable du tableau utilisée pour les ordonnées	Chaine de caractères correspondant à une variable du tableau	y=”taille”
ci	variable permettant de contrôler l’intervalle de confiance affiché	Entier entre 0 et 100.	ci=99
nboot	variable permettant d’indiquer le nombre de réechantillonage bootstrap réalisés.	Entier	nboot=100
seed	variable indiquant une graine pour le bootstrap. Permet la reproductibilité.	Entier	seed=42
logistic	Variable permettant de choisir de faire une régression logistique	Booléen	logistic=True
lowess	Variable permettant de choisir de faire une régression LOWESS.	Booléen	lowess=True
robust	Variable permettant de choisir de faire une régression robuste.	Booléen	robust=True

REGPLOT

Exemple

```
sns.regplot(data=data, x="bill_length_mm",y="bill_depth_mm", ci=70)
plt.show()
```






```
sns.regplot(data=data, x="bill_length_mm", y="bill_depth_mm",
 ci=99, lowess=True)
plt.show()
```

L'intervalle de confiance n'est pas affiché lorsque l'on utilise une lowess.

LMPLOT

Définition

seaborn.lmplot(**data**, *, **x=None**, **y=None**, **hue=None**, **col=None**, **row=None**, palette=None, col_wrap=None, height=5, aspect=1, markers='o', sharex=None, sharey=None, hue_order=None, col_order=None, row_order=None, legend=True, legend_out=None, x_estimator=None, x_bins=None, x_ci='ci', scatter=True, fit_reg=True, **ci=95**, **n_boot=1000**, units=None, seed=None, order=1, logistic=False, **lowess=False**, robust=False, logx=False, x_partial=None, y_partial=None, truncate=True, x_jitter=None, y_jitter=None, scatter_kws=None, line_kws=None, facet_kws=None)

LMPLOT

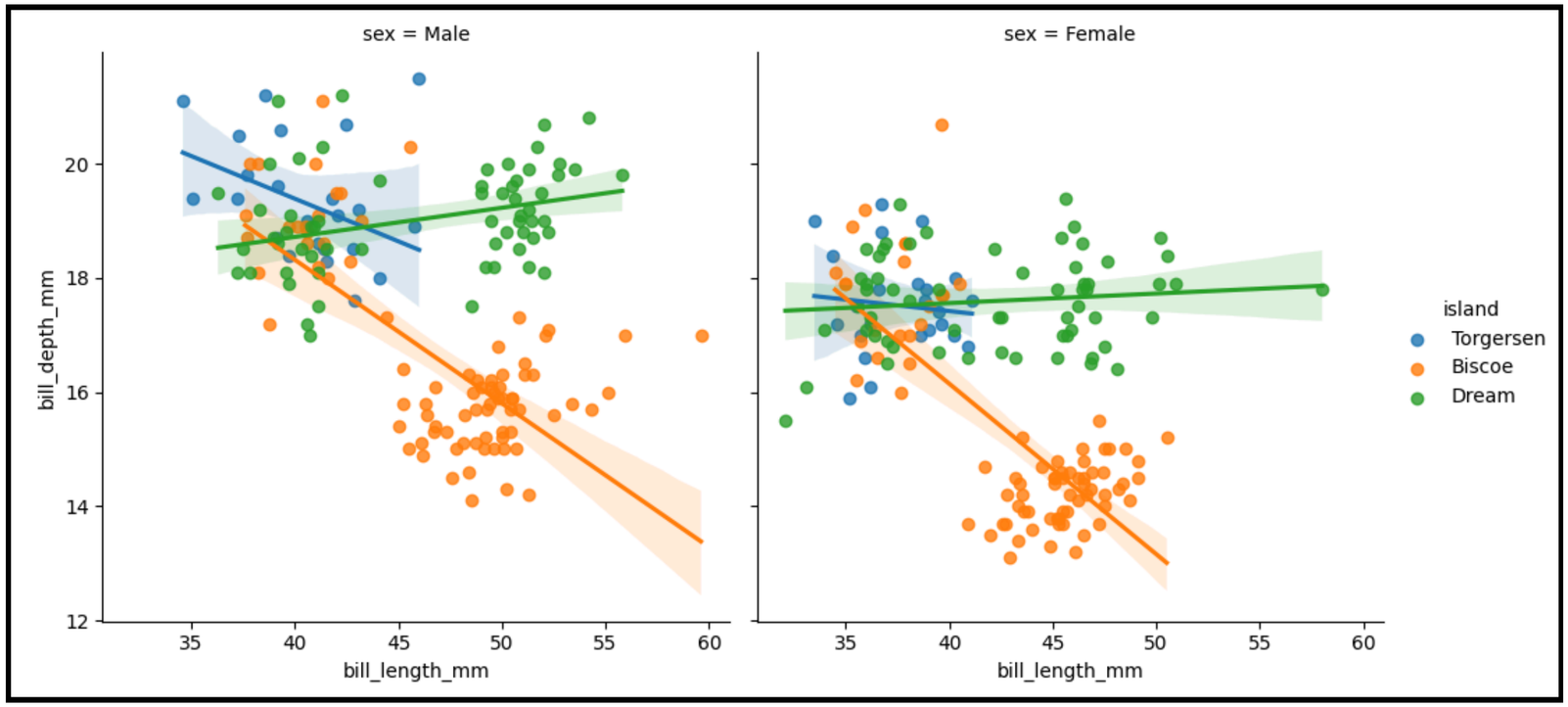
Paramètres

Nom du paramètre	Explications	Ce qui lui faut comme format	Exemple
data	Il faut donner au paramètre *data* le tableau entier que vous traitez	DataFrame, Series, dict, array, or list of arrays	data=tableau
x	variable du tableau utilisée pour les abscisses	Chaine de caractères correspondant à une variable du tableau	x="poids"
y	variable du tableau utilisée pour les ordonnées	Chaine de caractères correspondant à une variable du tableau	y=”taille”
hue	variable du tableau permettant de rajouter une dimension avec de la couleur	Chaine de caractères correspondant à une variable du tableau, catégorielle ou entière	hue=”age”
row	variable du tableau qui permettra de créer un tableau de graphiques, ici les lignes	Chaine de caractères correspondant à une variable du tableau, catégorielle ou entière	row=”catégorie”
col	variable du tableau qui permettra de créer un tableau de graphiques, ici les colonnes	Chaine de caractères correspondant à une variable du tableau, catégorielle ou entière	col=”métier”
ci	variable permettant de contrôler l’intervalle de confiance affiché	Entier entre 0 et 100.	ci=99
nboot	variable permettant d’indiquer le nombre de réechantillonage bootstrap réalisés.	Entier	nboot=100
lowess	Variable permettant de choisir de faire une régression LOWESS.	Booléen	lowess=True

LMPLOT

Exemple

```
sns.lmplot(data=data, x="bill_length_mm", y="bill_depth_mm", ci=95, hue="island", robust=True, col="sex")
plt.show()
```



Les régressions robust et logistic sont aussi disponibles comme pour regplot.
nboot et seed aussi.

HEATMAP

Définition

seaborn.heatmap(**data**, *, **vmin=None**, **vmax=None**, **cmap=None**, center=None, robust=False, **annot=None**, fmt='.2g', annot_kws=None, **linewidths=0**, **linecolor='white'**, cbar=True, cbar_kws=None, cbar_ax=None, square=False, xticklabels='auto', yticklabels='auto', **mask=None**, ax=None, **kwargs)

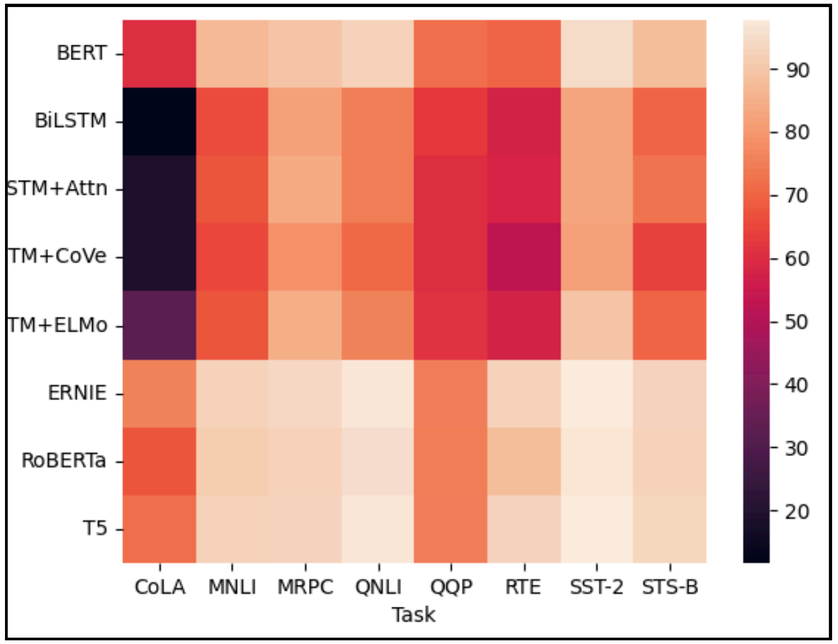
HEATMAP

Paramètres

Nom du paramètre	Explications	Ce qui lui faut comme format	Exemple
data	Il faut donner au paramètre *data* le tableau entier que vous traitez	DataFrame, Series, dict, array, or list of arrays	data=tableau
cmap	Couleurs de la heatmap. Soit une palette de matplotlib soit une personalisée.	Chaîne de caractères correspondant à une palette ou une color_palette de seaborn.	cmap=”viridis” ou cmap = sns.color_palette("light:blue", as_cmap=True)
annot	Variable qui choisit si on affiche les valeurs des cellules	Booléen	annot=True, vaut False par défaut
vmin	Valeur minimum qui sera prise en compte pour la colormap	Valeur flottante	vmin=30.6
vmax	Valeur maximale qui sera prise en compte pour la colormap	Valeur flottante	vmax=42
linecolor	Variable permettant de choisir la couleur des lignes entre les cellules.	Chaîne de caractère correspondant à une couleur	linecolor=”blue”
linewidths	Variable contrôlant l’épaisseur des lignes entre les cellules	Valeur flottante	linewidths=0.2 ou linewidths=10
mask	Variable permettant de contrôler les valeurs prises en compte dans la heatmap.	Tableau de booléen au même format que data.	mask=tableau_mask

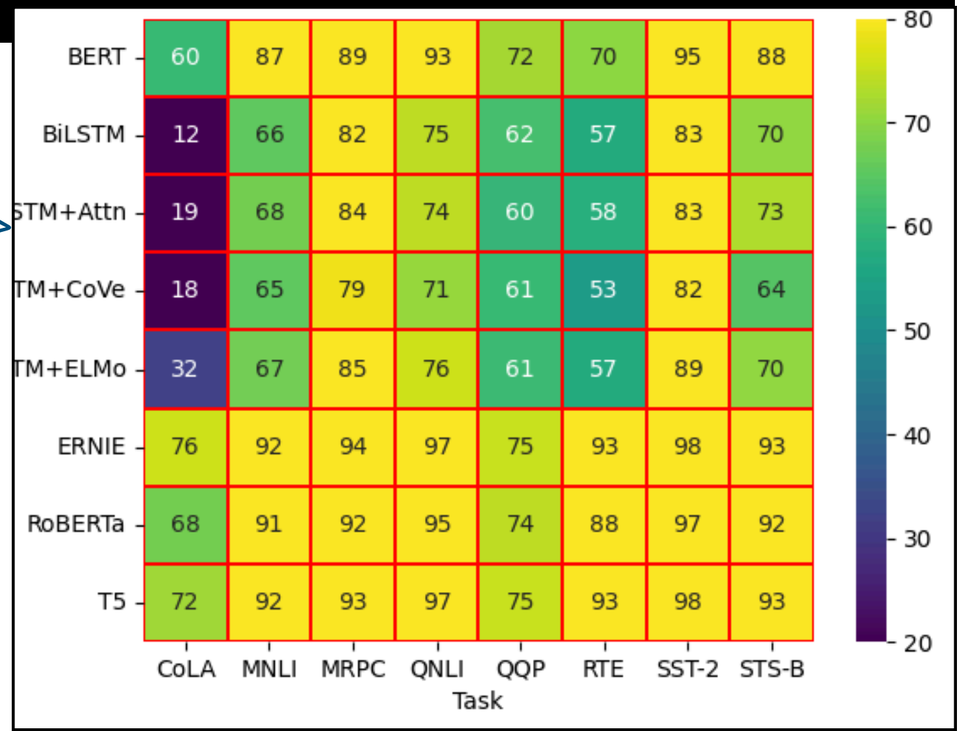
HEATMAP

Exemple



pivot() permet de formater le dataset dans le format adéquat aux heatmaps et donc de choisir les axes et les valeurs. Chaque combinaison d'*index* et *columns* doit avoir une valeur.

```
glue=sns.load_dataset("glue").pivot(index="Model",columns="Task",values="Score")
sns.heatmap(glue)
```




```
sns.heatmap(glue,cmap="viridis",annot=True,vmin=20,vmax=80,linecolor="red",linewidths=0.1)
```

CLUSTERMAP

Définition

Nécessite scipy

seaborn.clustermap(**data**, *, pivot_kws=None, **method='average'**, **metric='euclidean'**, **z_score=None**, **standard_scale=None**, **figsize=(10, 10)**, cbar_kws=None, **row_cluster=True**, **col_cluster=True**, row_linkage=None, col_linkage=None, row_colors=None, col_colors=None, mask=None, **dendrogram_ratio=0.2**, colors_ratio=0.03, **cbar_pos=(0.02, 0.8, 0.05, 0.18)**, tree_kws=None, **kwargs)

CLUSTERMAP

Paramètres

Nom du paramètre	Explications	Ce qui lui faut comme format	Exemple
data	Il faut donner au paramètre *data* le tableau entier que vous traitez	DataFrame, Series, dict, array, or list of arrays	data=tableau
method	Méthode scipy pour faire le clustering	Chaîne de caractère correspondant à une méthode de scipy	method='centroid'
metric	Métrique scipy utilisée pour faire le clustering	Chaîne de caractère correspondant à une métrique de scipy	metric='jaccard'
z_score	Variable permettant de centrer et réduire les données.	0 pour centrer et réduire les lignes, 1 pour les colonnes	z_score=0
standard_scale	Variable permettant de normaliser les données.	0 pour normaliser les lignes, 1 pour les colonnes	standard_scale=1
row_cluster,col_cluster	Variables permettant de choisir les axes de clustering	Booléen	row_cluster=False, faut True par défaut
figsize	Variable contrôlant la taille de la figure	tuple(largeur,hauteur)	figsize=(4,4)
dendrogram_ratio	Variable contrôlant le ratio de taille des dendogram	tuple(ratio de ligne, ratio de colonne)	dendrogram_ratio=(0.2,0.1)
cbar_pos	Variable contrôlant la position de la barre de couleur.	tuple(gauche,bas,largeur,hauteur)	cbar_pos=(0,0.1,0.05,0.6)

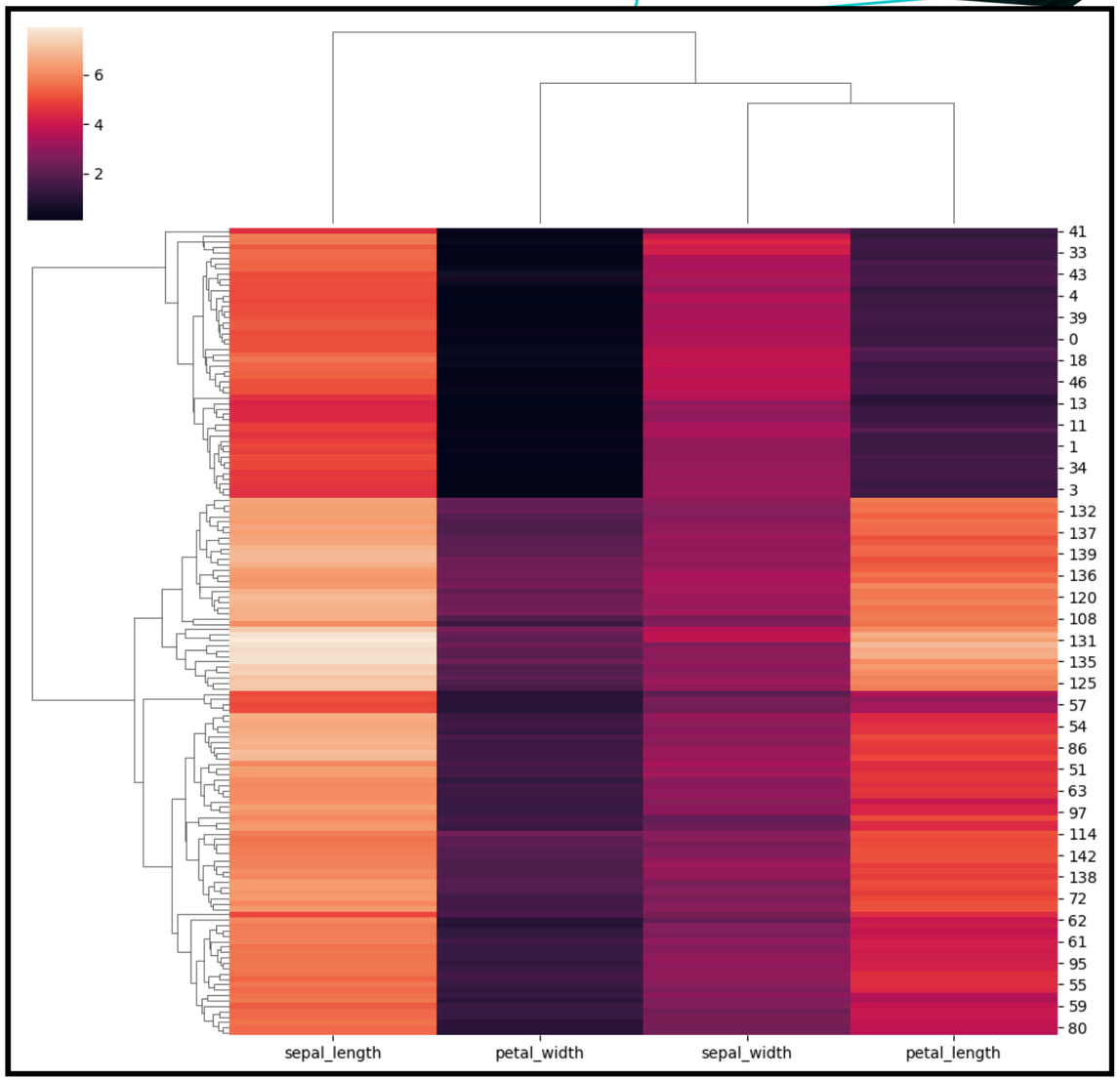
CLUSTERMAP

Exemple

```
iris = sns.load_dataset("iris")
species = iris.pop("species")
sns.clustermap(iris)
```

On a les dendrogrammes sur les 4 métriques, donc on sait par exemple que sepal_width et petal_length sont proches. On a aussi les dendrogrammes sur les individus.

Le clustering peut dépendre des méthodes et des métriques utilisées.

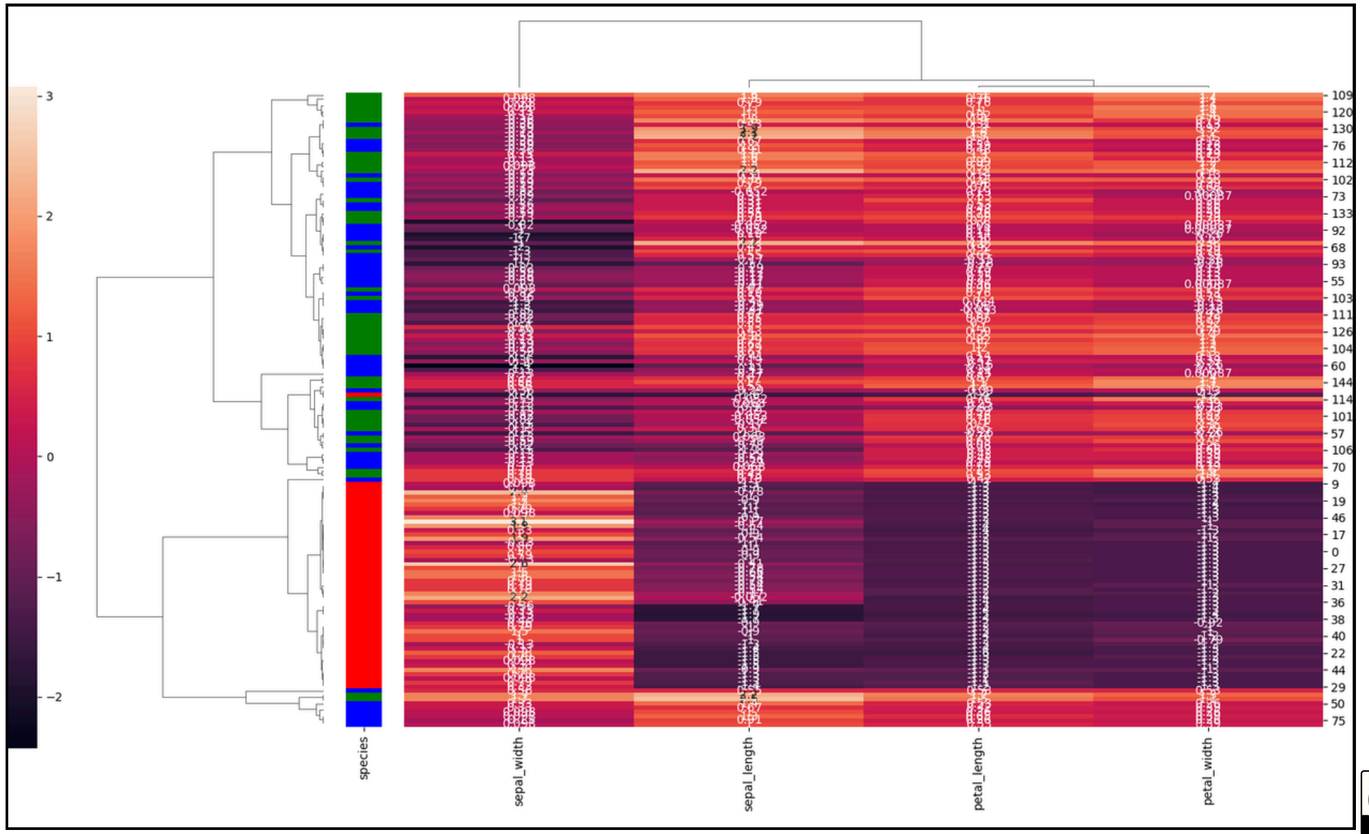


CLUSTERMAP

Exemple

```
lut = dict(zip(species.unique(), "rbg"))
row_colors = species.map(lut)
sns.clustermap(iris,row_cluster=True,dendrogram_ratio=(0.2,0.1),row_colors=row_colors,method='weighted',metric='correlation',z_core=1
,annot=True,figsize=(3,9),cbar_pos=(0,0.1,0.02,0.8))
```

Ici en utilisant *row_colors* qui correspond aux espèces on peut voir par exemple que l'espèce rouge est très regroupée dans le clustering.



DONNÉES QUALITATIVES

LES DONNÉES QUALITATIVES

Que sont des données qualitatives ?

Les données qualitatives, ou catégorielles, sont, par opposition aux données quantitatives, toutes les données autres que des nombres. On peut retrouver des chaines de caractères (pour des noms par exemple) ou des booléens (True/False, oui/non, 1/0, ...).

Les données qualitatives sont tout à fait plotables avec les outils vus précédemment. Mais il existe un outil tout-en-un qui permet de passer d'un style à l'autre facilement.

FONCTION SEABORN

Cet outil c’est la fonction seaborn :

sns.catplot()

Il est possible d’utiliser format de plot de votre choix comme vu pour les données quantitatives (bar, box, violin, …), il faudra le préciser dans un des paramètres de catplot.

DÉFINITION COMPLÈTE DE LA FONCTION

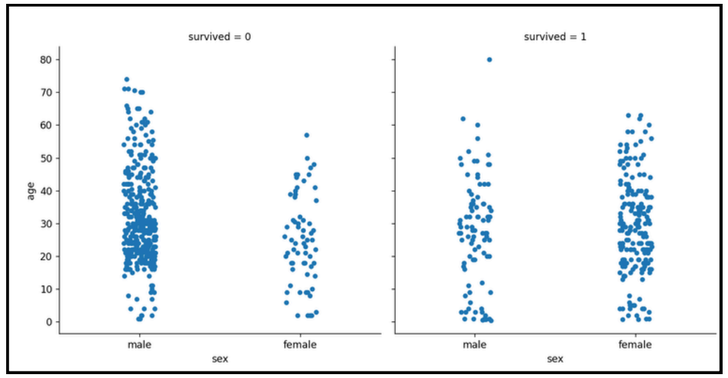
seaborn.catplot(data=None, *, x=None, y=None, hue=None, row=None, col=None, **kind='strip'**, estimator='mean', errorbar=('ci', 95), n_boot=1000, seed=None, units=None, weights=None, order=None, hue_order=None, row_order=None, col_order=None, col_wrap=None, height=5, aspect=1, log_scale=None, native_scale=False, formatter=None, orient=None, color=None, palette=None, hue_norm=None, legend='auto', legend_out=True, sharex=True, sharey=True, margin_titles=False, facet_kws=None, ci=<deprecated>, ****kwargs**)

ISDM INSTITUT DE SCIENCE DES DONNÉES MONTPELLIER

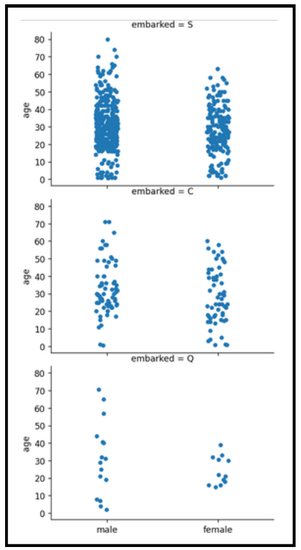
*Pour plus d'information, retrouvez la documentation **ici***

Explication des paramètres

Nom du paramètre	Explications	Ce qui lui faut comme format	Exemple
row, col	Ces deux paramètres permettent de séparer les données avec des paramètres supplémentaires que juste x et y. row et col, permettent de séparer les données dans différents graphes juxtaposés. Donc dans le même exemple, si on écrit col='alive', on va avoir deux colonnes ('yes', 'no') dans lesquelles on aura 'age'=f('sex'). Pareil pour row, mais ce sera trois lignes.	Nom d'une colonne de votre tableau data ou si vous n'avez pas de tableau, directement le vecteur de données	col = "alive" row = "embarked"



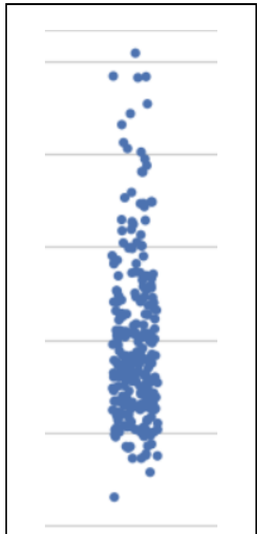

```
sns.catplot(data = tableau, x="sex", y="age)
```



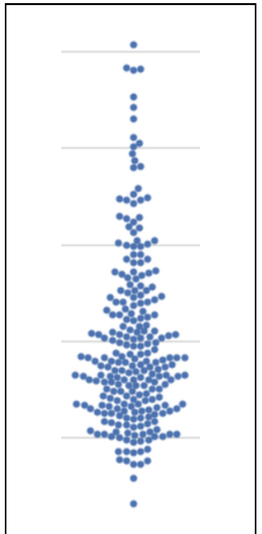

```
sns.catplot(data = tableau, x="sex", y="age)
```

Explication des paramètres

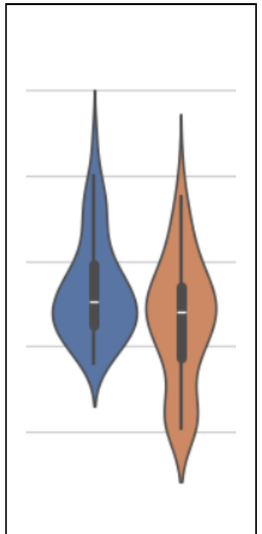
Nom du paramètre	Explications	Ce qui lui faut comme format	Exemple
kind	Permet de spécifier sous quel format afficher les données. Par exemple au-dessus, c'était kind='strip'. Il y a 8 possibilités : "strip", "swarm", "box", "violin", "boxen", "point", "bar", or "count"	Il y a 8 possibilités : "strip", "swarm", "box", "violin", "boxen", "point", "bar", or "count"	kind="violin"



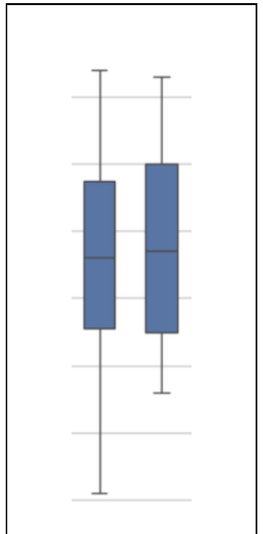
"strip"



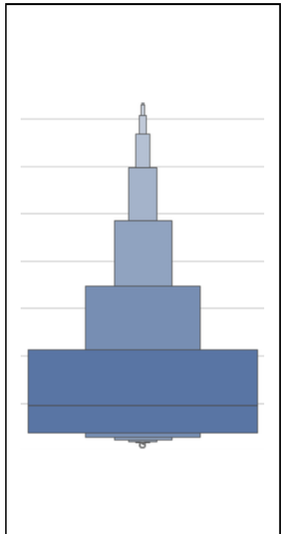
"swarm"



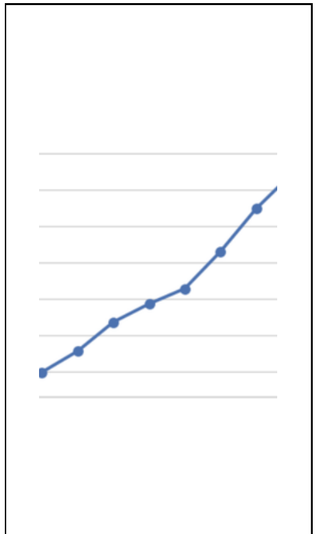
"violin"



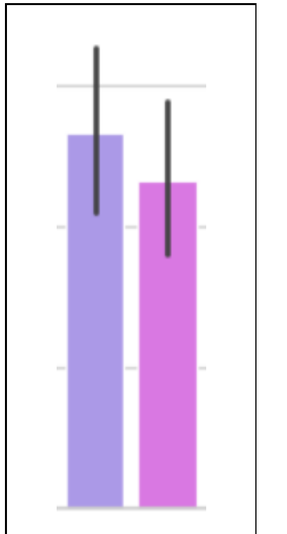
"box"



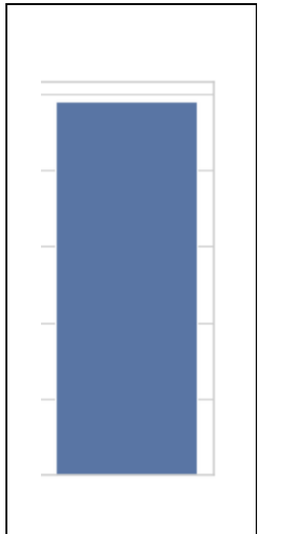
"boxen"



"point"



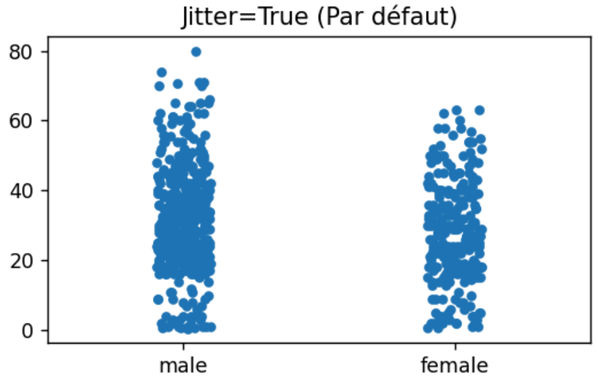
"bar"

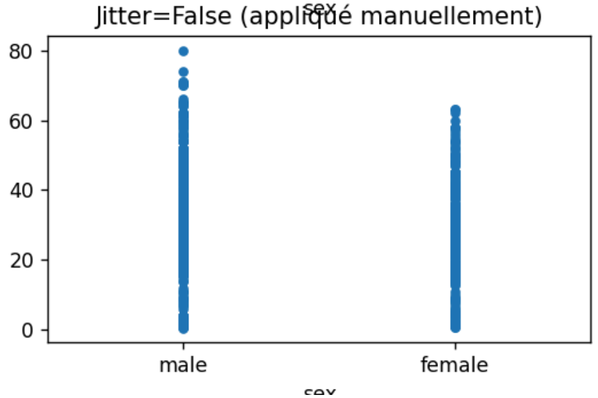


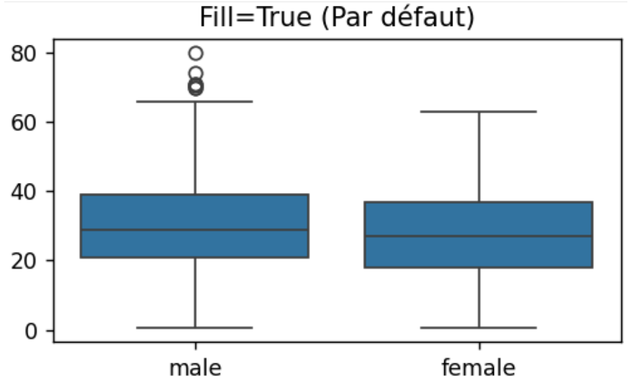
"count"

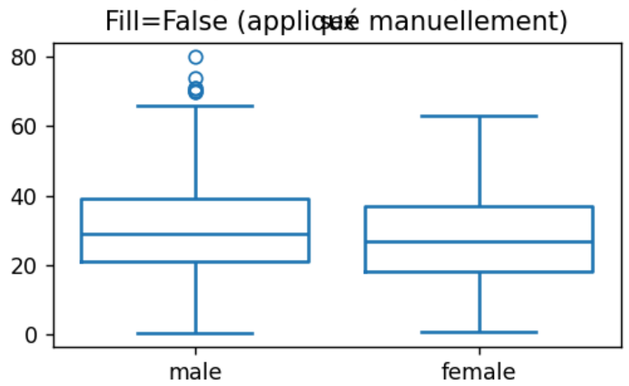
Explication des paramètres

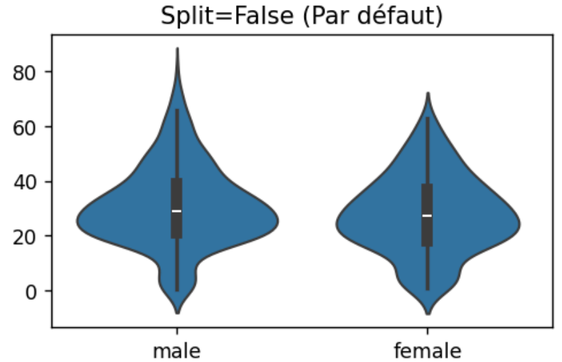
Nom du paramètre	Explications	Ce qui lui faut comme format	Exemple
kwargs	Permet de rajouter des paramètres propre à la méthode d'affichage choisie, donc des paramètres que vous retrouverez dans la documentation propre à chaque méthode. Par exemple, pour la méthode d'affichage "strip", les données sont « étalées » autour de l'axe de l'abscisse pour une question de lisibilité, mais c'est possible de l'enlever (en passant le paramètre jitter (paramètre dans la fonction stripplot()) à faux : jitter=False)	Nom d'une colonne de votre tableau data ou si vous n'avez pas de tableau, directement le vecteur de données	stripplot() : jitter = False boxplot() : fill = False violinplot() : split = True

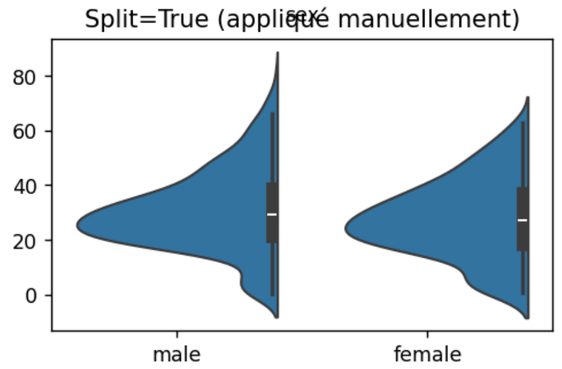












REPRÉSENTATIONS SPÉCIALES

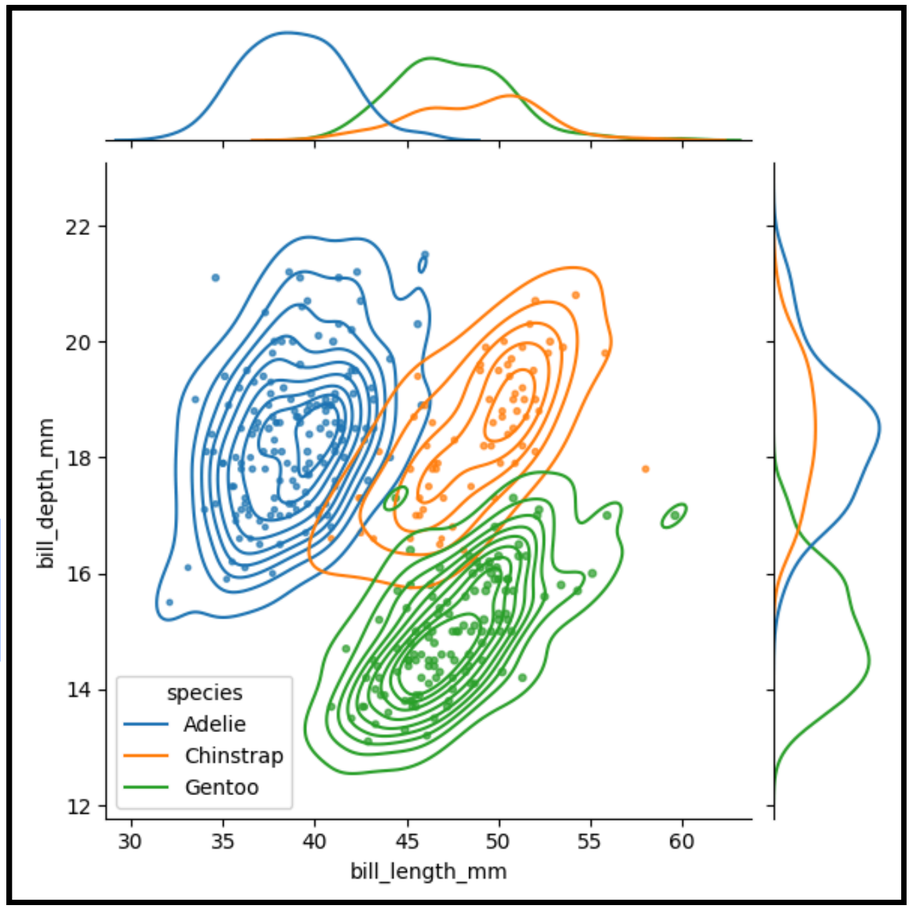
JOINTPLOT

Avec seaborn on peut facilement afficher des graphiques bidimensionnels et les graphiques monodimensionnel correspondants :

```
g=sns.jointplot(data=data, x="bill_length_mm", y="bill_depth_mm",
 kind="kde", hue="species")

for species in data["species"].unique():
    subset = data[data["species"] == species]
    g.ax_joint.scatter(
        subset["bill_length_mm"],
        subset["bill_depth_mm"],
        s=subset["body_mass_g"]/500,
        label=species,
        alpha=0.7)
```

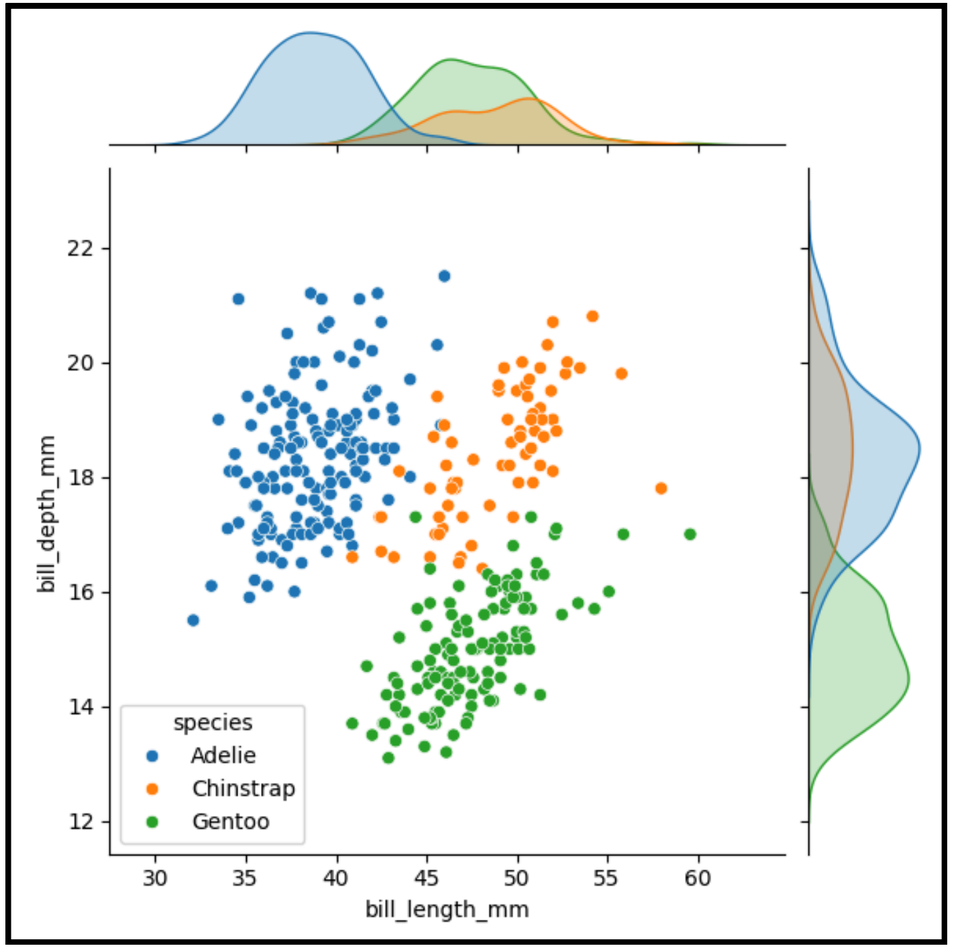
Code matplotlib pour l'affichage des points en plus des kde



JOINTGRID

JointGrid comparé à joinplot permet d'avoir plus de contrôle et de personnalisation.
On peut notamment utiliser des types de graphique différents entre le bidimensionnel et les monodimensionnels.

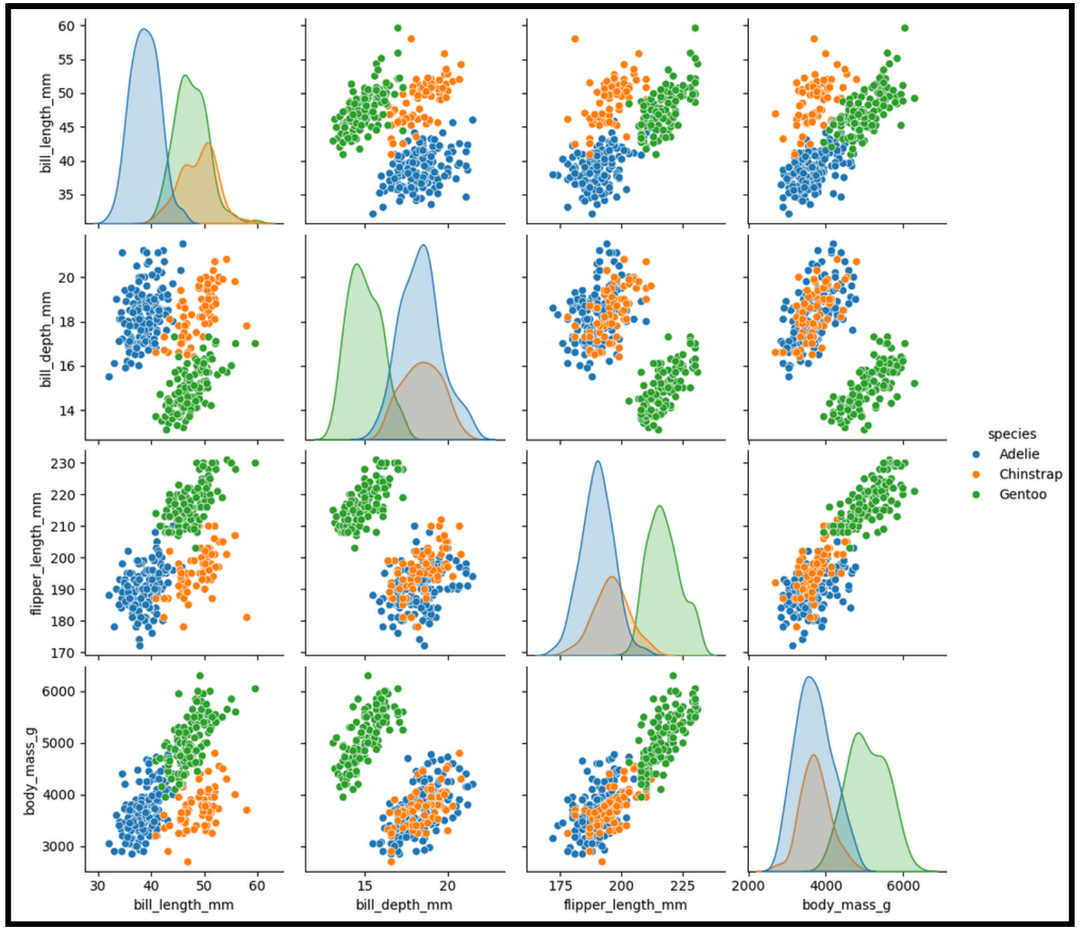
```
g5 = sns.JointGrid(data=data, x="bill_length_mm", y="bill_depth_mm",
 hue="species")
g5.plot_joint(sns.scatterplot)
g5.plot_marginals(sns.kdeplot, fill=True)
```



PAIRPLOT

Pairplot permet d’utiliser toutes les variables numériques du tableau afin de faire des correspondances 2 à 2.

```
sns.pairplot(data=data, hue=”species”)
```

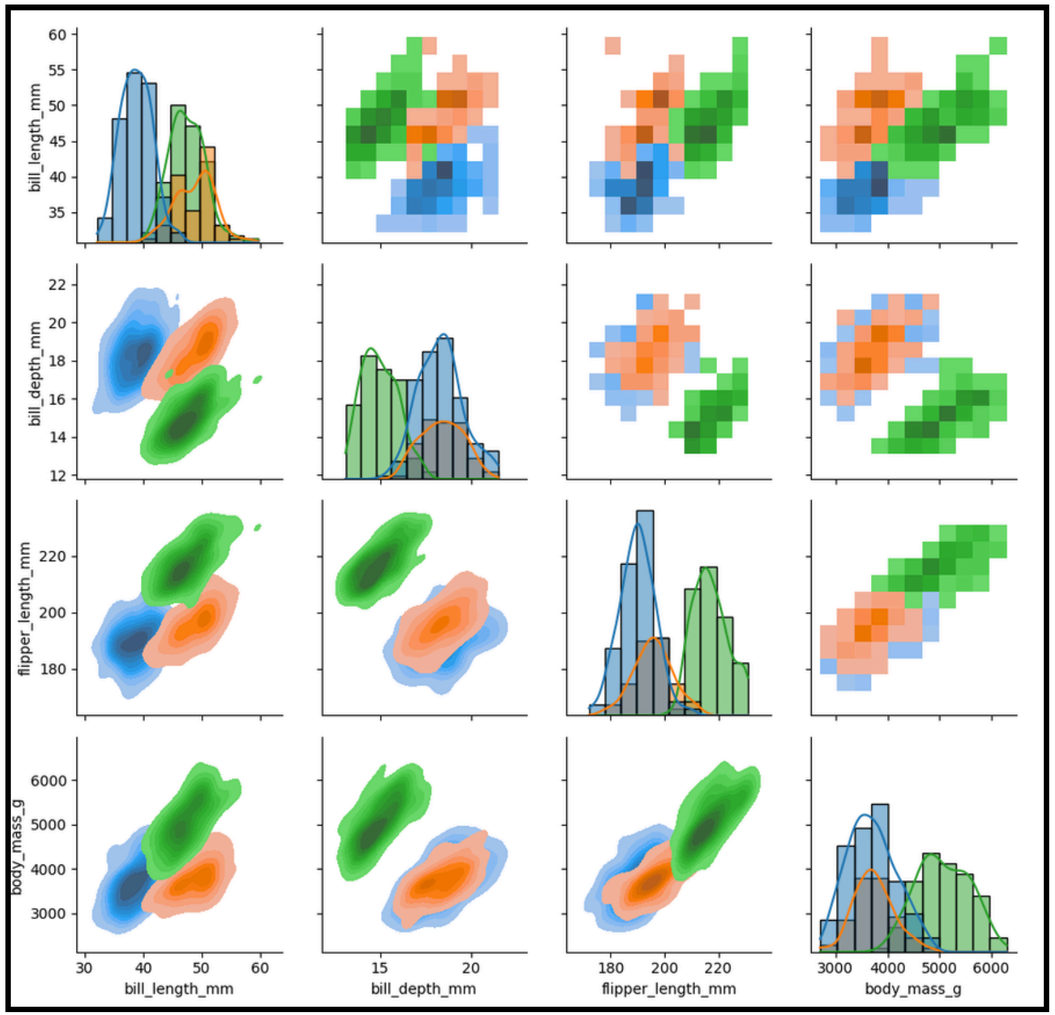


PAIRGRID

Même logique que pour les jointplot et JointGrid.

```
g7 = sns.PairGrid(data, hue="species")
g7.map_upper(sns.histplot)
g7.map_lower(sns.kdeplot, fill=True)
g7.map_diag(sns.histplot, kde=True)
```

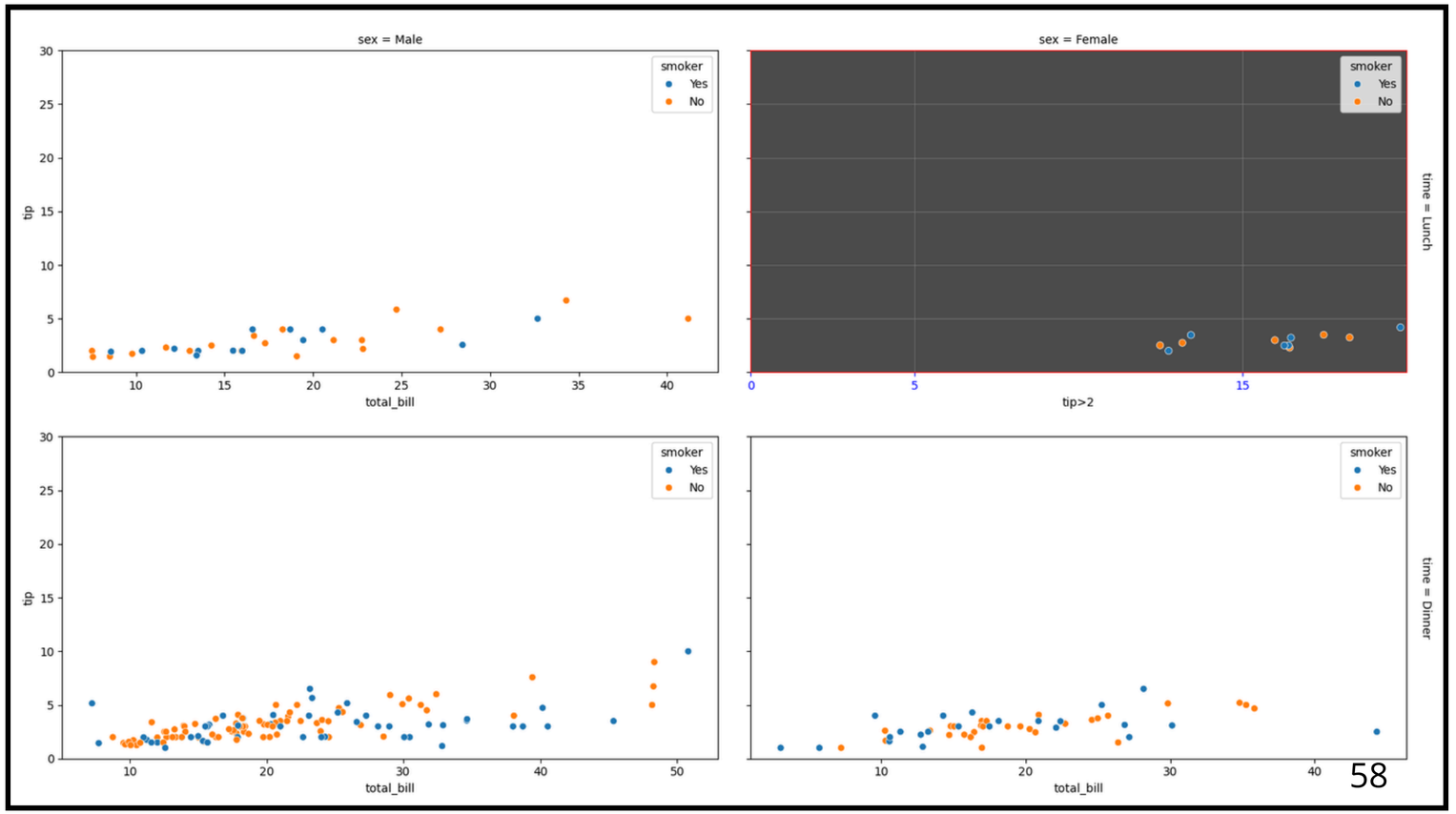
On peut chosir les différents graphiques pour la diagonale, partie supérieure et partie inférieure.



FACETGRID

FacetGrid est l'outil permettant de complètement contrôler une grille de graphique, pemettant facilement de traiter des sous-ensemble de données dans une même structure.

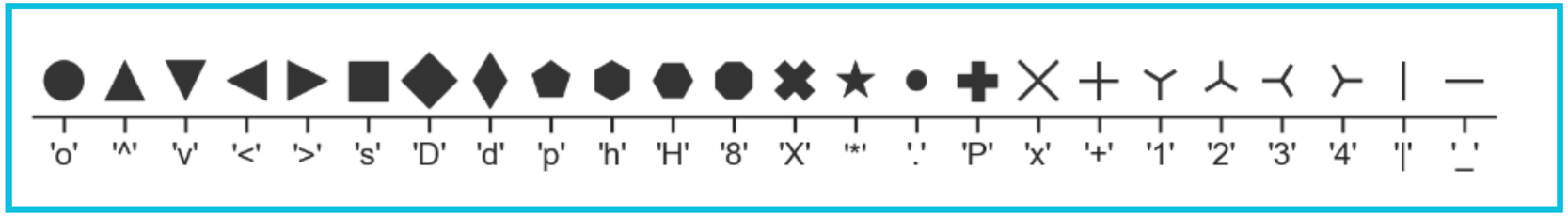
```
tips = sns.load_dataset("tips")
g = sns.FacetGrid(tips,col="sex",row="time",margin_titles=True,
    despine=False,sharex=False)
g.figure.subplots_adjust(wspace=0.05, hspace=0.2)
for (row_val, col_val), ax in g.axes_dict.items():
    subset = tips[(tips["time"] == row_val) & (tips["sex"] == col_val)]
    if row_val == "Lunch" and col_val == "Female":
        subset = subset[subset["tip"] > 2]
        sns.scatterplot(data=subset,x="total_bill",y="tip",hue="smoker",
        ax=ax)
        ax.set_facecolor(".3")
        ax.set_xlim(0,20)
        ax.set_ylim(0,30)
        ax.set_xticks([0, 5, 15])
        ax.set_xlabel("tip>2")
        ax.grid(True, color="gray", linestyle="-", linewidth=0.5)
        ax.spines["top"].set_color("red")
        ax.spines["right"].set_color("red")
        ax.spines["bottom"].set_color("red")
        ax.spines["left"].set_color("red")
        ax.tick_params(axis="x", colors="blue")
    else:
        ax.set_facecolor((0, 0, 0, 0))
        sns.scatterplot(data=subset,x="total_bill",y="tip",hue="smoker",
        ax=ax)
```



LA CUSTOMISATION DANS SEABORN

LES MARKERS

Les markers existants



Les markers à construire

Polygones = **0**
(nb_cotés, **0**, angle)

Étoile = **1**
(nb_branches, **1**, angle)

Astérix = **2**
(nb_branches, **2**, angle)

LIGNES POINTILLÉES

Format	Example
À l'aide de tiret	'-' —————————— '- -' - - - - - - - - - - - - - - - - - - '-.' —·—·—·—·—·—·—·—·—·—·—·—· ':' .....
En précisant le schéma voulu sous forme de tuple (largeur_tiret, largeur_vide, largeur_tiret, largeur_vide, ...)	(6, 2) — — — — — — — — — — — — — — — (2, 1) ---------- (0.5, 0.5) .....
	(4, 1, 2, 1) -·-·-·-·-·-·-·-·-·-·-·-·-

LES COULEURS

Paramètre ***color***

Format	Example
RGB or RGBA (red, green, blue, alpha) tuple de valeurs décimales dans l'intervalle [0, 1].	color=(0.1, 0.2, 0.5) ou color=(0.1, 0.2, 0.5, 0.3)
RGB or RGBA en hexadécimal (insensible à la casse)	color='#0f0f0f' ou color='#0f0f0f80'
RGB or RGBA en hexadécimal (insensible à la casse) abrégé lors de double caractères	'#abc' as '#aabbcc' ou '#fb1' as '#ffbb11'
Valeur décimale entre 0 et 1 pour des nuances de gris	'0' as black, '1' as white, '0.8' as light gray
Pour les couleurs classiques, une lettre suffit.	'b' as blue, 'g' as green, 'r' as red, 'c' as cyan, 'm' as magenta, 'y' as yellow, 'k' as black, 'w' as white. color='m'
Accès aux couleurs prédéfinies ***X11/CSS4*** (sans espace) (insensible à la casse). Contient 148 couleurs (peut varier).	https://fr.wikipedia.org/wiki/Noms_de_couleur_X11
Accès aux couleurs prédéfinies ***xkcd*** (insensible à la casse). Utiliser le préfixe 'xkcd:nom_couleur'. Contient 954 couleurs.	https://xkcd.com/color/rgb/
Couleurs du tableau 'T10' (insensible à la casse). Nom des couleurs : blue, orange, green, red, purple, brown, pink, gray, olive, cyan. Syntaxe : 'tab:nom_couleur' *Note : ce sont les couleurs par défault. C'est-à-dire que si vous avez une couleur elle sera 'tab:blue', et si vous en avez plusieurs, elles seront prises dans l'ordre donc d'abord orange puis vert, etc.*	

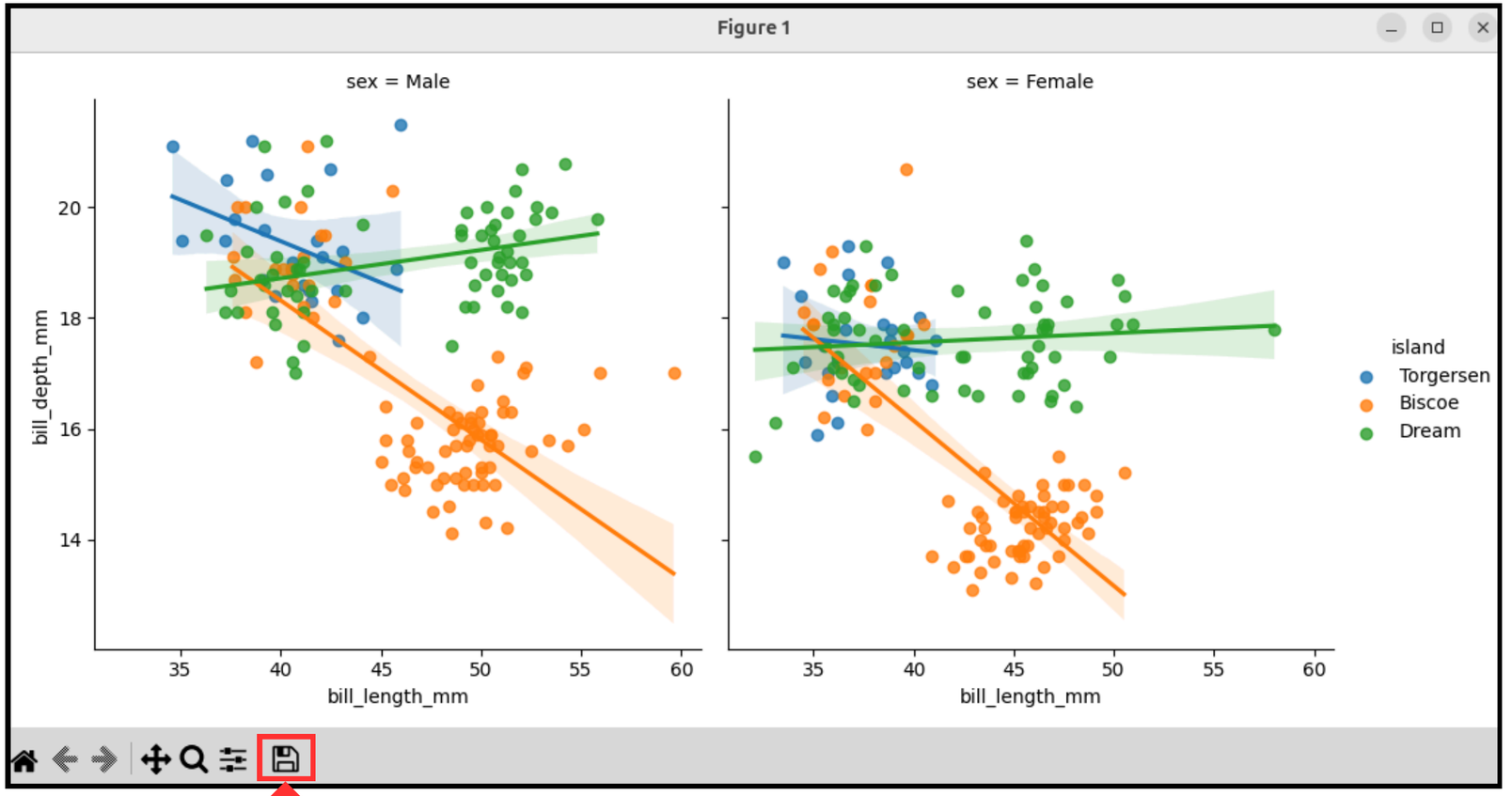
LES COULEURS

Paramètre ***color_palette***

Format	**Example**
Palette de couleurs discrètes déjà existantes	https://www.practicalpythonfordatascience.com/ap_seaborn_palette
(Demi-) palette de couleurs continue déjà existantes. Ces palettes vont de couleurs très claires à des couleurs “foncées”. Elles servent à visualiser des données dont l’importance est graduelle. Les données les moins intéressantes en clair et les données pertinentes en foncé.	"rocket", "mako", "flare", and "crest" (sachant que les deux dernières sont semblables aux deux premières respectivement mais inversées) (si ces palettes sont utilisées sur des données discrètes, elles seront discrètisées automatiquement).  
Palette de couleurs continues déjà existantes. Ces palettes sont “complètes”, c’est-à-dire que toutes les données seront colorés avec la même intensité mais pas la même couleur. À utiliser dans un cadre où toutes les données sont pertinentes.	“vlag” et “icefire”  
Beaucoup plus d’information sont disponibles sur la documentation en ligne. Notamment pour la librairie matplotlib (que Seaborn utilise), il y a cette page qui rassemble beaucoup d’information : https://matplotlib.org/stable/users/explain/colors/colormaps.html	

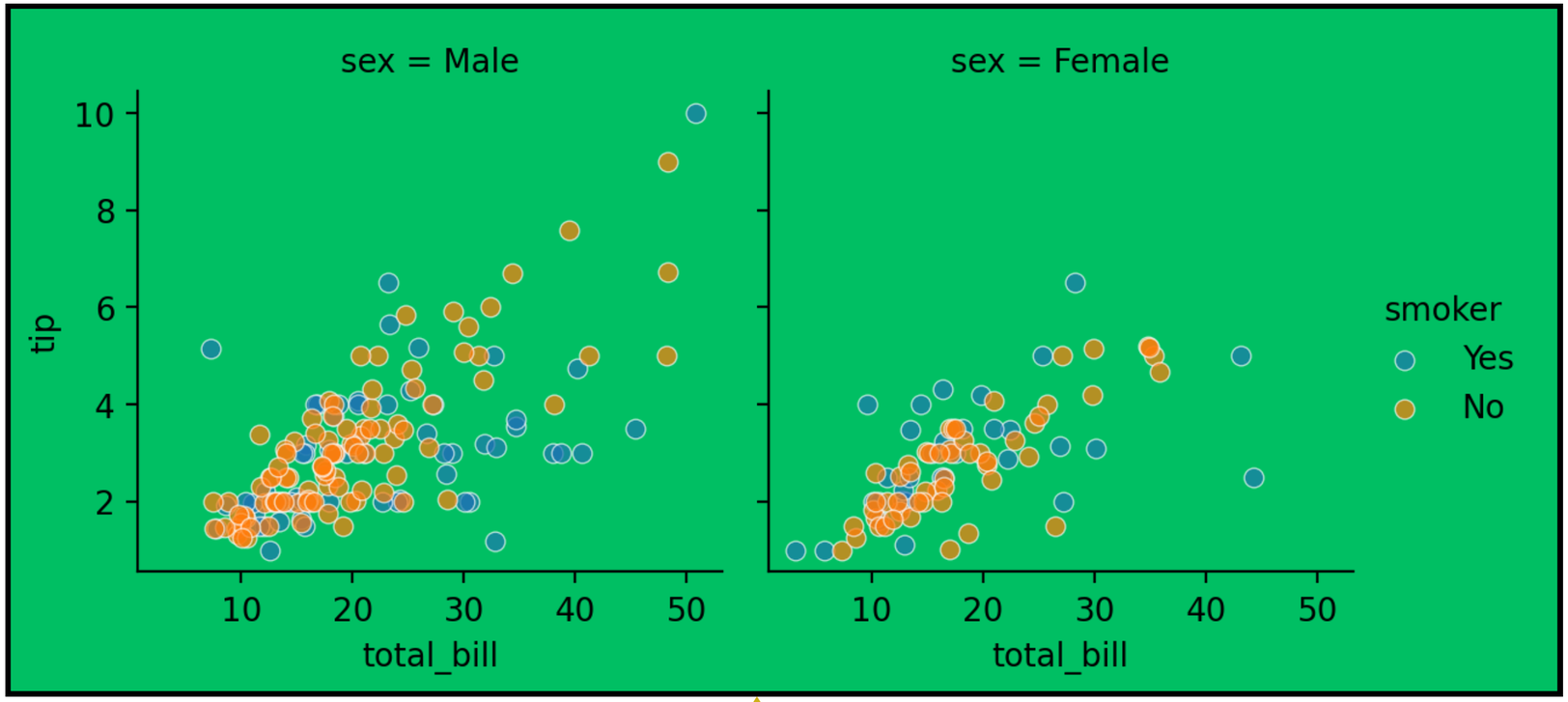
SAUVEGARDE DE FIGURES

LE BOUTON DE SAUVEGARDE



Vous pourrez sauvegarder ou vous voulez mais vous n'aurez pas d'options

LA FONCTION SAVEFIG



Fond vert d'un site par exemple

Si votre graphique est la variable g :

```
g.savefig("img",dpi=200,transparent=True)
```

Le dpi est la résolution en dot par inch. Plus ce paramètre est faible plus l'image sera petite et de mauvaise qualité.

transparent permet de retirer le fond de l'image pour pouvoir mettre la figure sur des fonds de différentes couleurs.

OBJETS SEABORN

LES OBJETS SEABORN

Depuis la mise à jour 0.12 de Seaborn des objets seaborn ont été introduits. Ceux-ci constituent une alternative puissante aux fonctions de plot originelles. Les objects sont inspirées de ggplot2 de R.

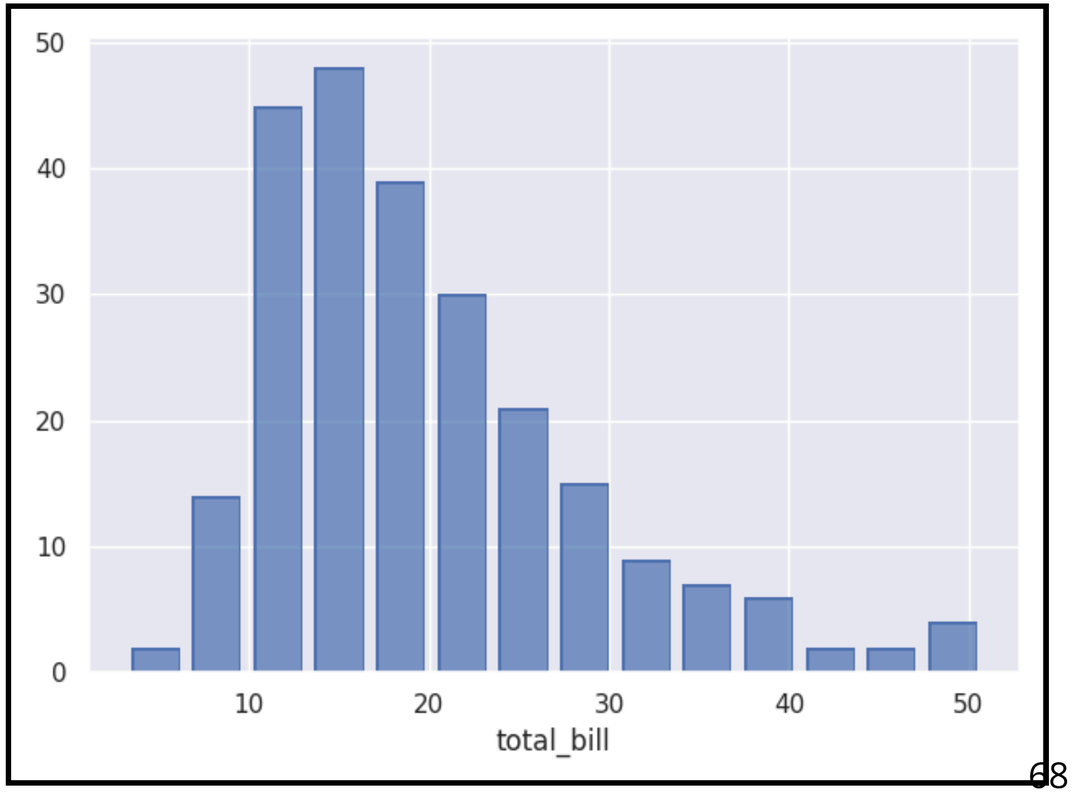
```
import seaborn.objects as so
```

Et une seule fonction permet de faire des graphiques : `so.Plot()`

A ce plot on peut donner des données : `so.Plot(tips,x="total_bill")`

Et à ces données on rajoute les objets graphiques :

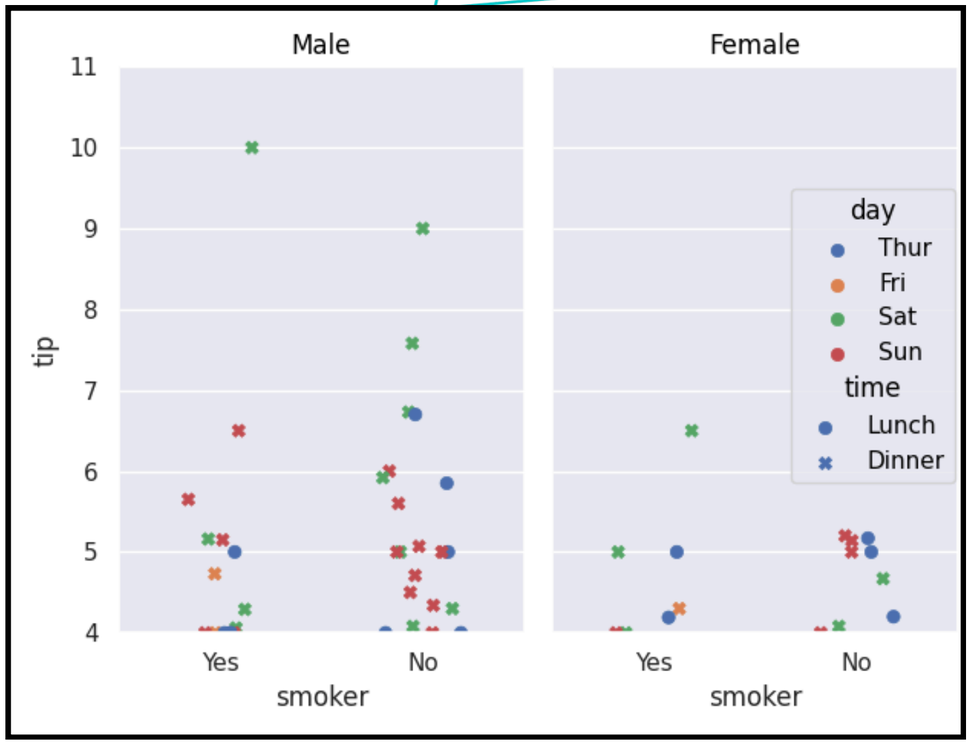
```
so.Plot(tips,x="total_bill").add(so.Bar(),so.Hist()).show()
```

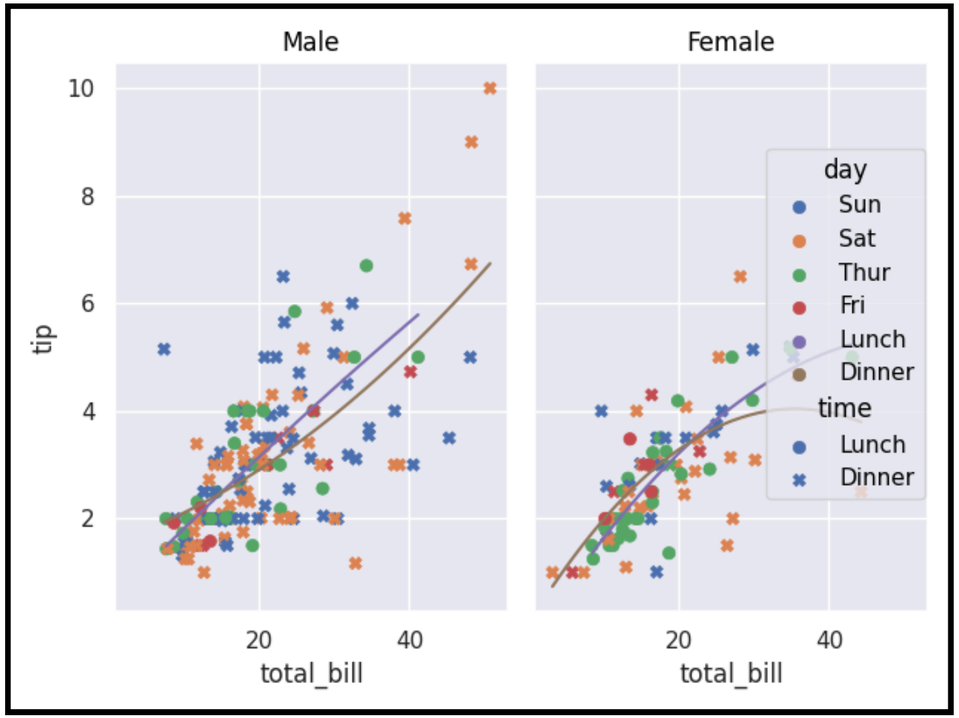


DOT

L'équivalent du scatterplot en Seaborn objet est le Dot() que l'on ajoute à un Plot pour lequel on a définit les données. L'équivalent de *hue* est *color* et *facet* est celui de *row* et *col* que l'on peut spécifier dans facet.

```
so.Plot(tips,x="smoker",y="tip").add(so.Dot(),so.Jitter(),color="day",
marker="time").facet("sex").limit(y=(4,11)).show()
```






```
so.Plot(tips,x="total_bill",y="tip").add(so.Dot(),color="day",marker="time").
facet("sex").add(so.Line(),so.Polyfit(),group="time",color="time").show()
```

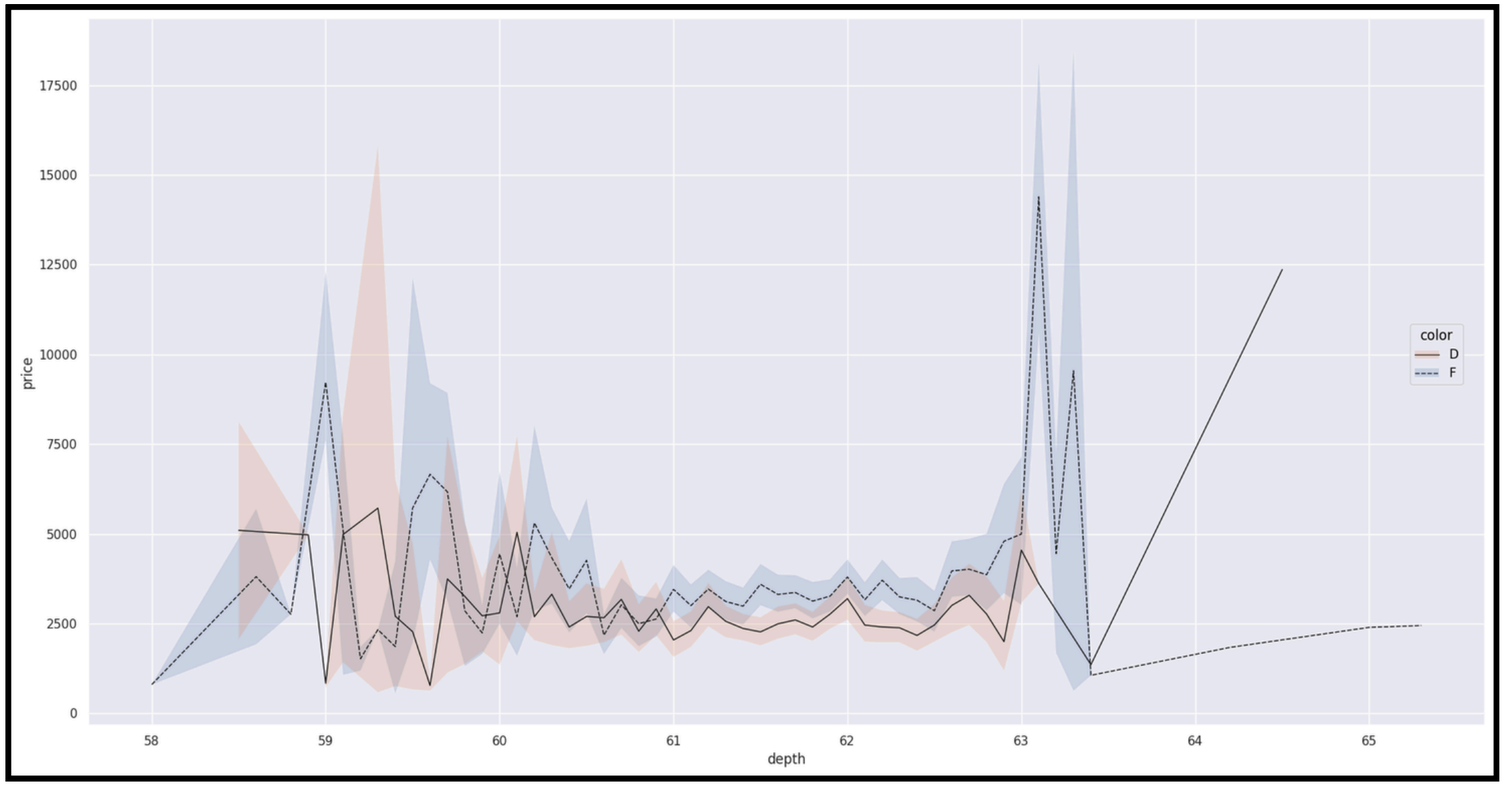
L'objet Line() permet de faire des scatterplot de type ligne, et l'on peut même spécifier si l'on veut une régression et son type.

LINE

```
diamonds.query("cut == 'Ideal' and color != 'J' and color!= 'E' and color!='H' and color!='I' and color!='G'").pipe(so.Plot, "depth", "price",linestyle="color").add(so.Line(color=".1",linewidth=1),so.Agg()).add(so.Band(), so.Est(),group="color",color="color").show()
```

Nous avons vu l'objet Line() précédemment avec PolyFit(), ici nous utilisons Agg() qui permet de faire l'aggrégation des données selon les valeurs *price* ici. On peut choisir la méthode d'aggrégation si l'on veut.

Nous utilisons query() de pandas afin de ne travailler seulement sur un sous-ensemble de données. On ajoute aussi les objets Band() et Est() qui ensemble permettent d'afficher l'incertitude.

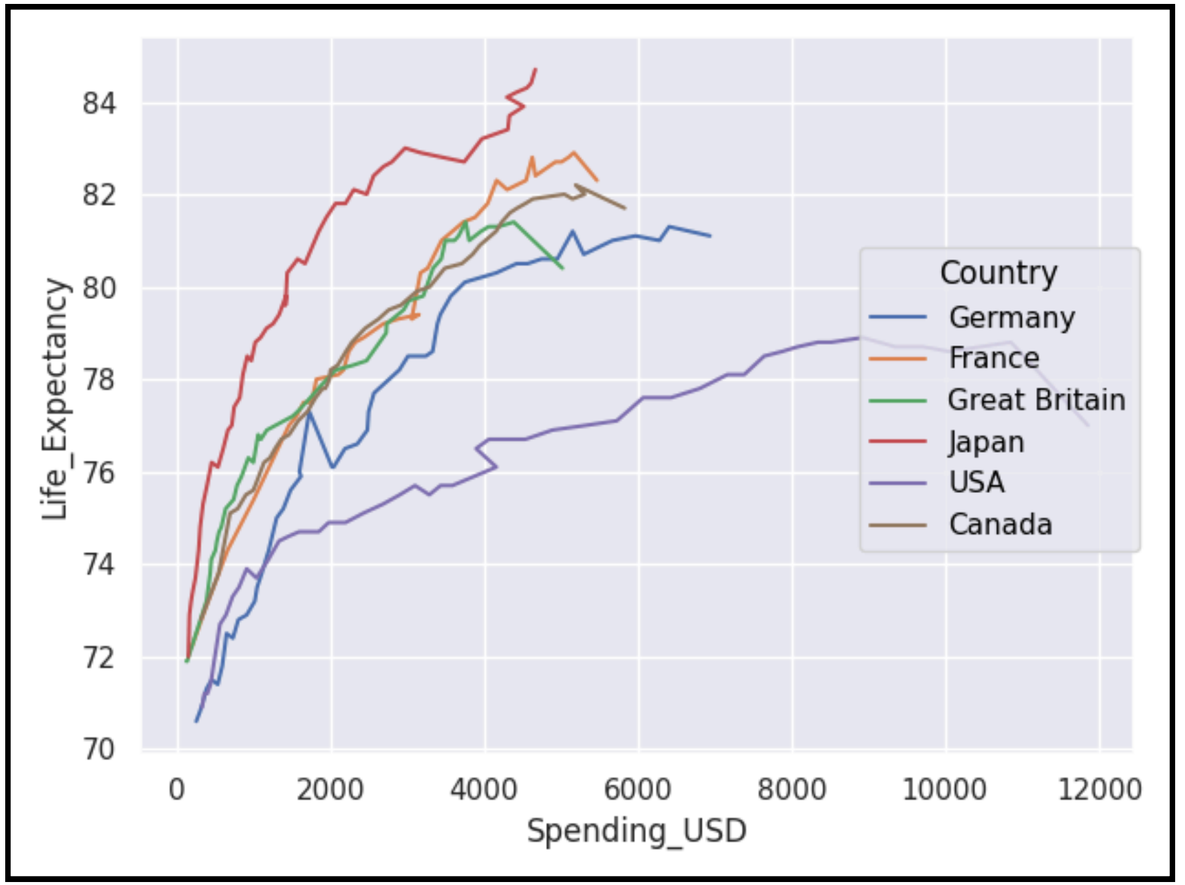


PATH

```
p = so.Plot(healthexp, "Spending_USD", "Life_Expectancy", color="Country").add(so.Path()).show()
```

Path() est un objet similaire à Line avec beaucoup de paramètres en commun.
La différence entre Path() et Line() vient du fait que Path() lira toujours les données dans l'ordre dans lequel elles sont écrites.

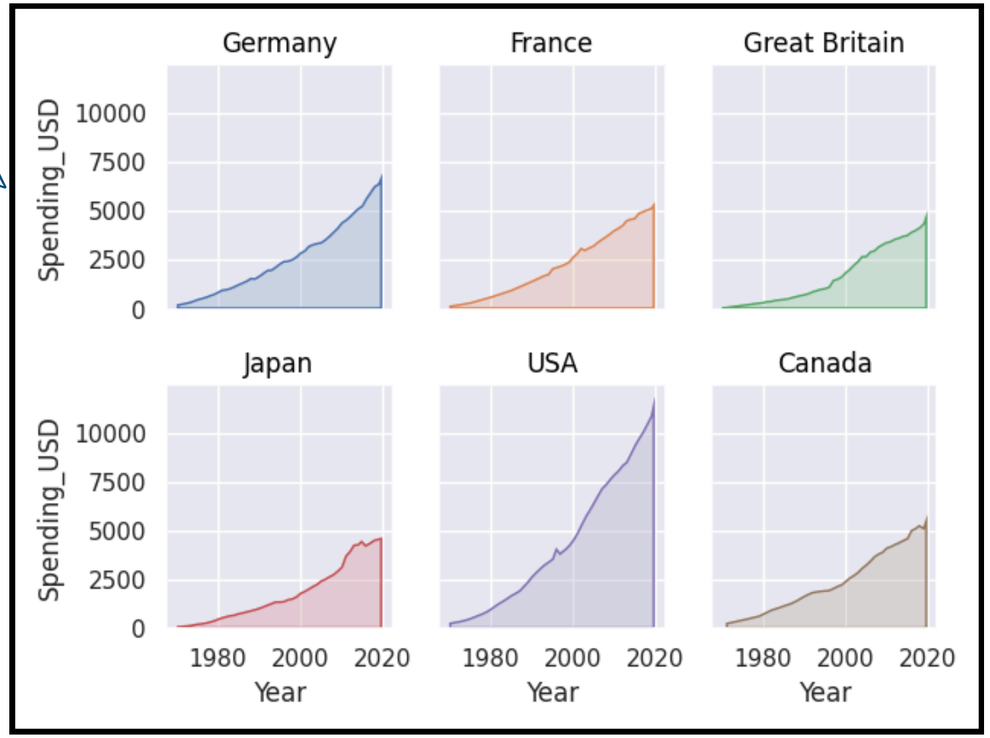
Cela convient bien pour visualiser des trajectoires.

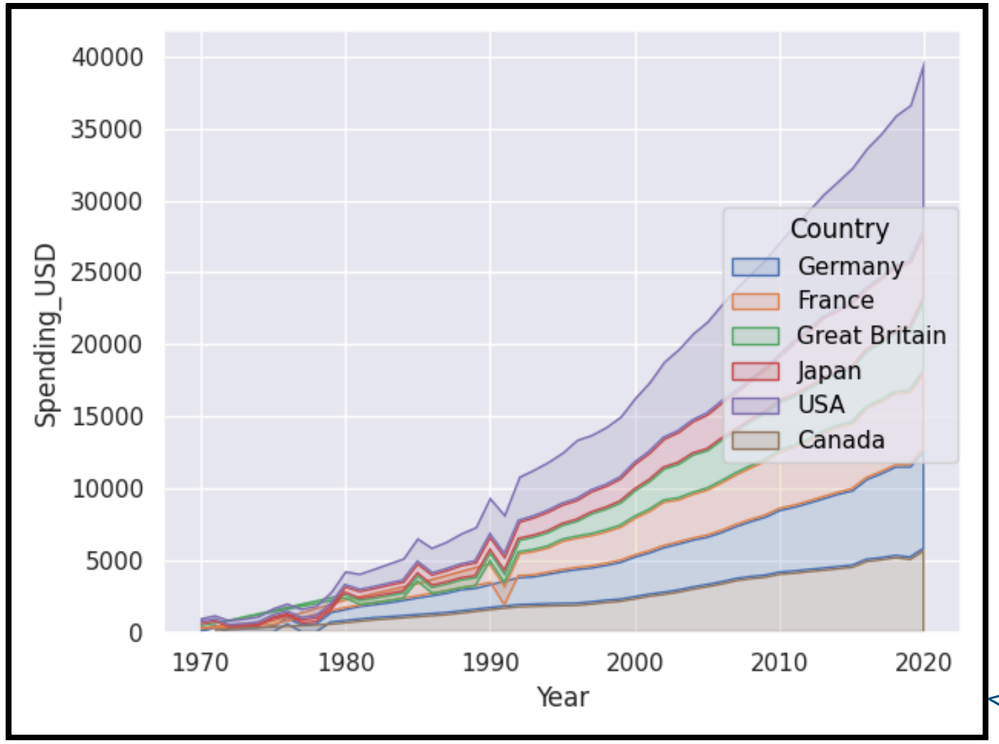


AREA

L’objet Area() permet de faire afficher des surfaces sous des courbes.

```
so.Plot(healthexp,"Year","Spending_USD").facet("Country",wrap=3).add(so.Area(),
color="Country",legend=False).show()
```





On peut rajouter Stack() afin d’empiler les surfaces.

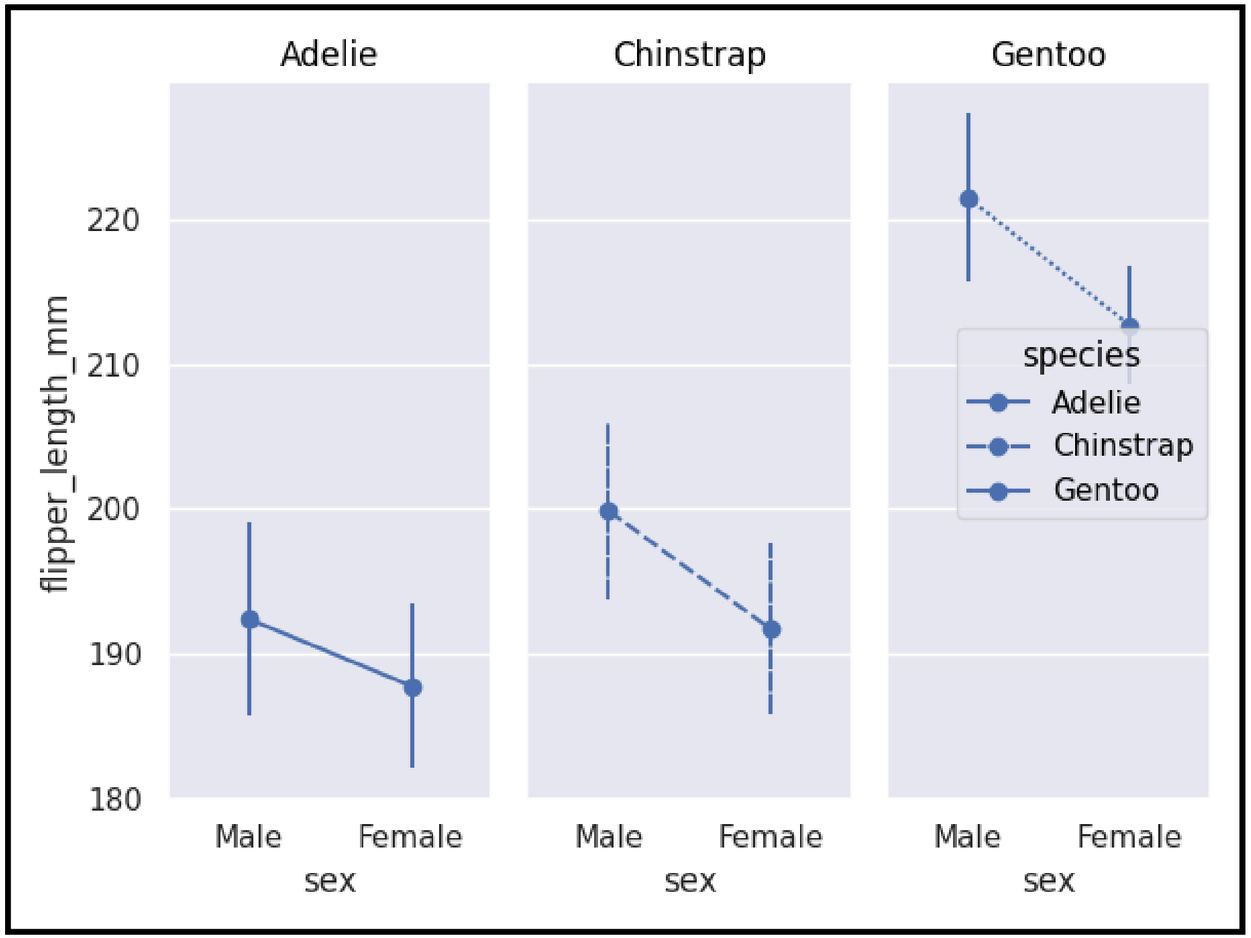
```
so.Plot(healthexp,"Year","Spending_USD",color="Country").add(so.Area(),
so.Stack()).show()
```

RANGE

```
so.Plot(peng,x="sex",y="flipper_length_mm",linestyle="species").
facet("species").add(so.Line(marker="o"), so.Agg())
 .add(so.Range(), so.Est(errorbar=("sd",1))).show()
```

L’objet Range() permet d’afficher des barres d’erreur, comparé à Band() qui permet d’afficher une surface.

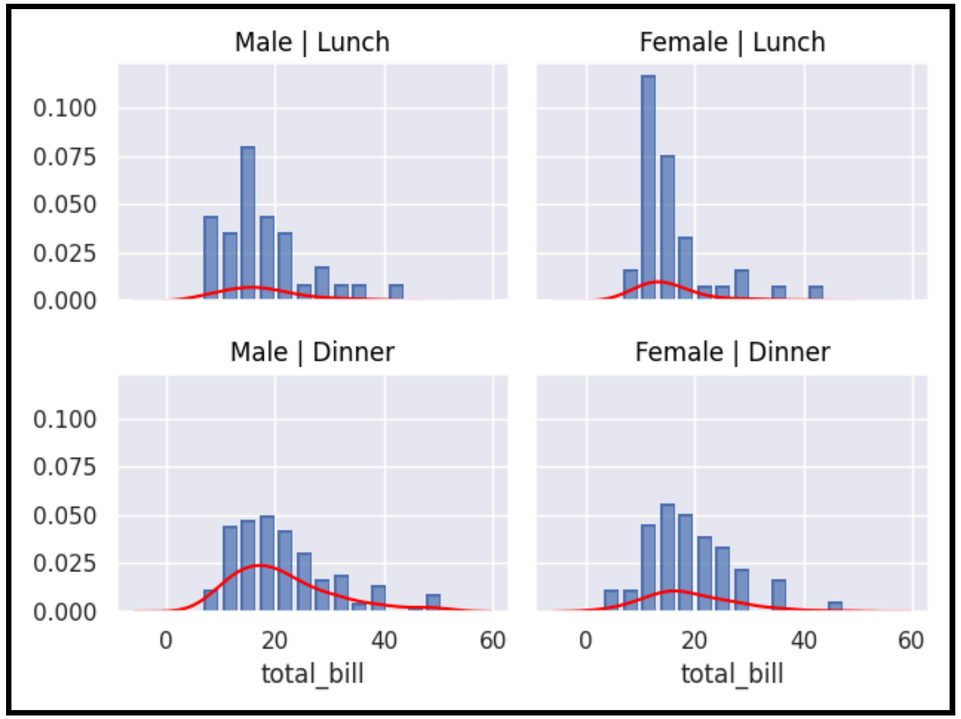
Nous pouvons choisir comment est calculée la barre d’erreur, ici on utilise l’écart type, c’est-à-dire que le minimum de la barre affiche la moyenne moins l’écart-type et la valeur maximum la moyenne plus l’écart-type. En utlisant *errorbar*=(“sd”,x) on peut contrôler combien d’écart-type on utilise.



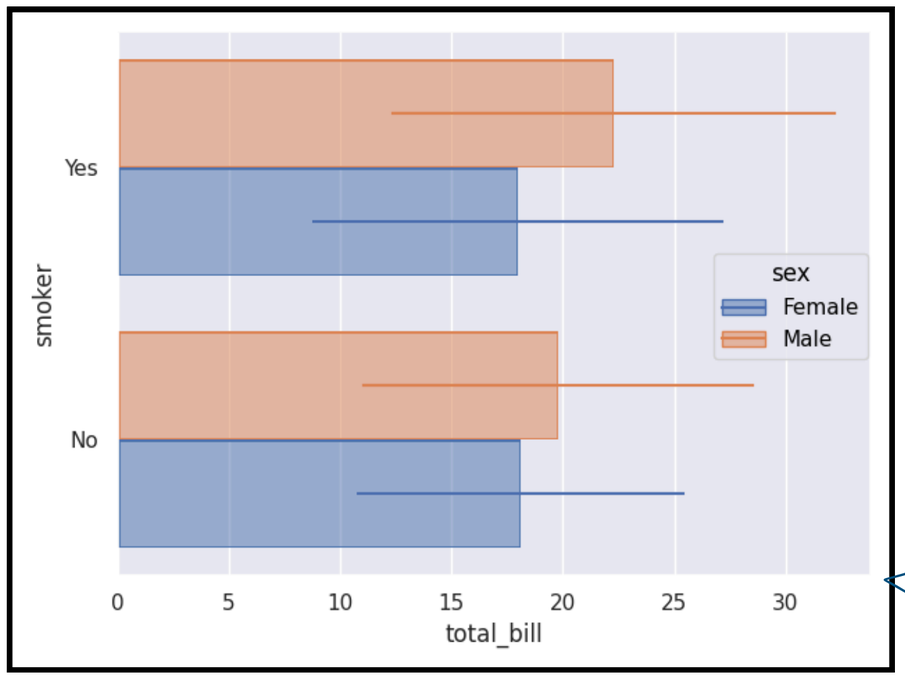
BAR

L'objet Bar() permet de faire des histogrammes. On peut contrôler ce que l'histogramme Hist() affiche avec stat.

```
so.Plot(tips,x="total_bill").facet(row="time",col="sex").add(so.Bar(),so.Hist(stat="density")).add(so.Line(color="red"),so.KDE()).show()
```



On voit ici une autre utilisation de Line(), on peut représenter des KDE avec l'objet KDE(). Si on fait une KDE sur 2 axes on a une carte de densité.



On peut utiliser l'objet Dodge() permet de séparer les données qui pourraient se superposer.

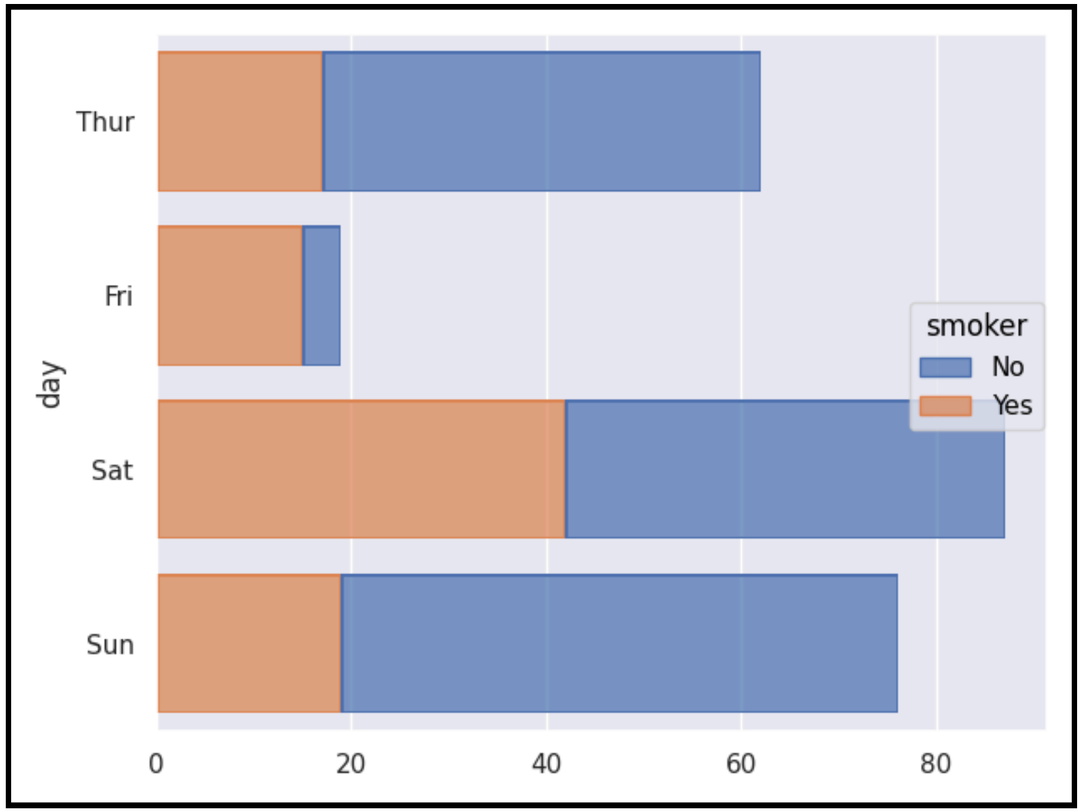
```
so.Plot(tips, "total_bill", "smoker", color="sex").add(so.Bar(alpha=.5), so.Agg(), so.Dodge()).add(so.Range(), so.Est(errorbar="sd"), so.Dodge()).show()
```

COUNT

L’objet Count() s’associe aussi avec Bar() pour faire des graphiques. Utilisé avec seulement un axe de renseigné il permet de compter le nombre d’occurence de certaines valeurs.

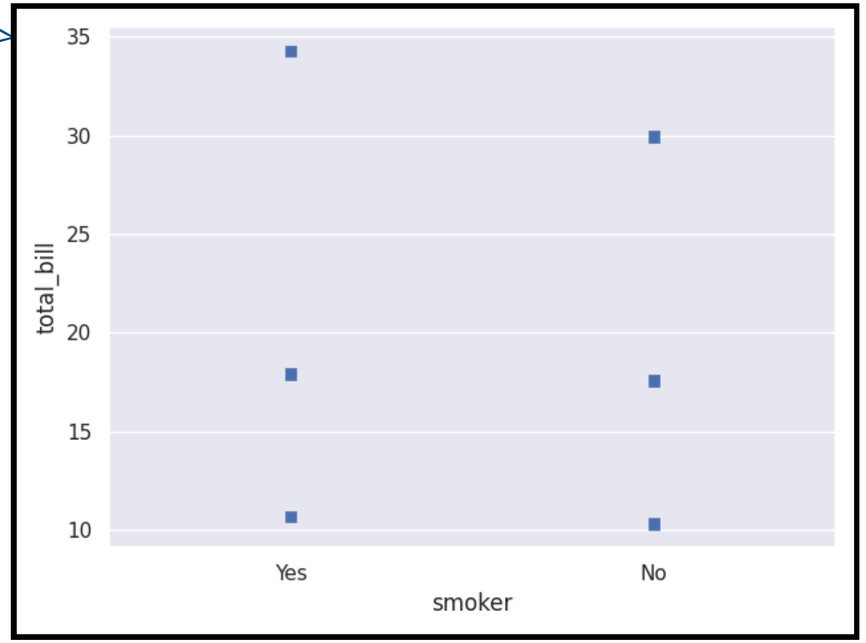
Nous pouvous soit utiliser Dodge() pour séparer les barres issues de color, soit Stack() pour les empiler.

```
so.Plot(tips,y="day",color="smoker").add(so.Bar(),so.Count(),so.Stack()).show()
```

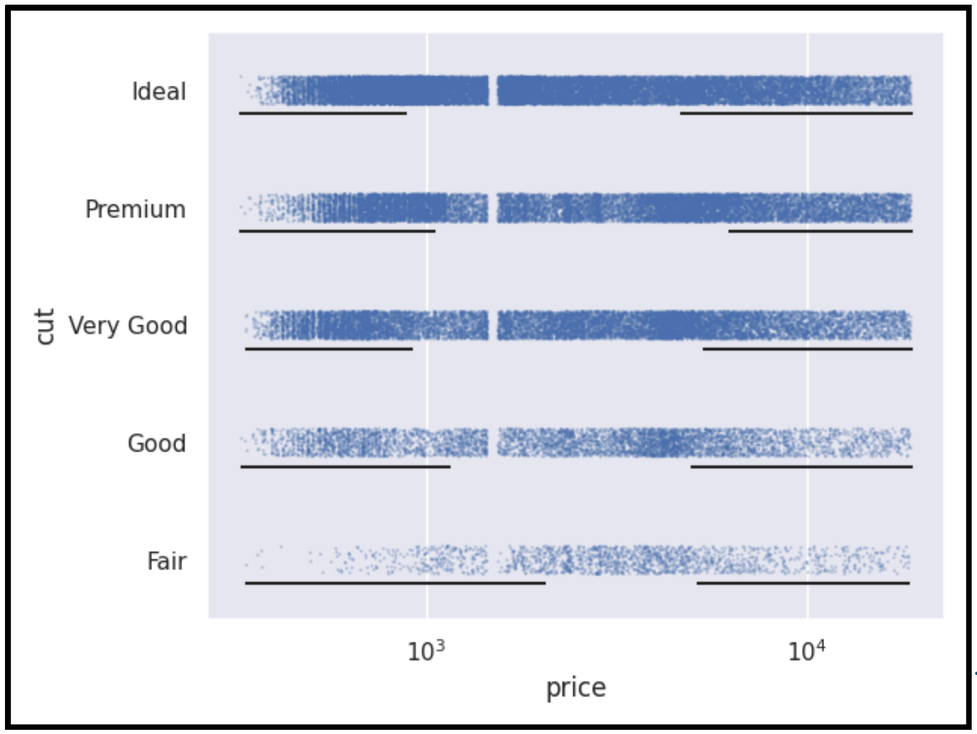


PERC

```
so.Plot(tips,"smoker","total_bill").add(so.Dot(marker="s"),so.Perc([10,50,90])).show()
```



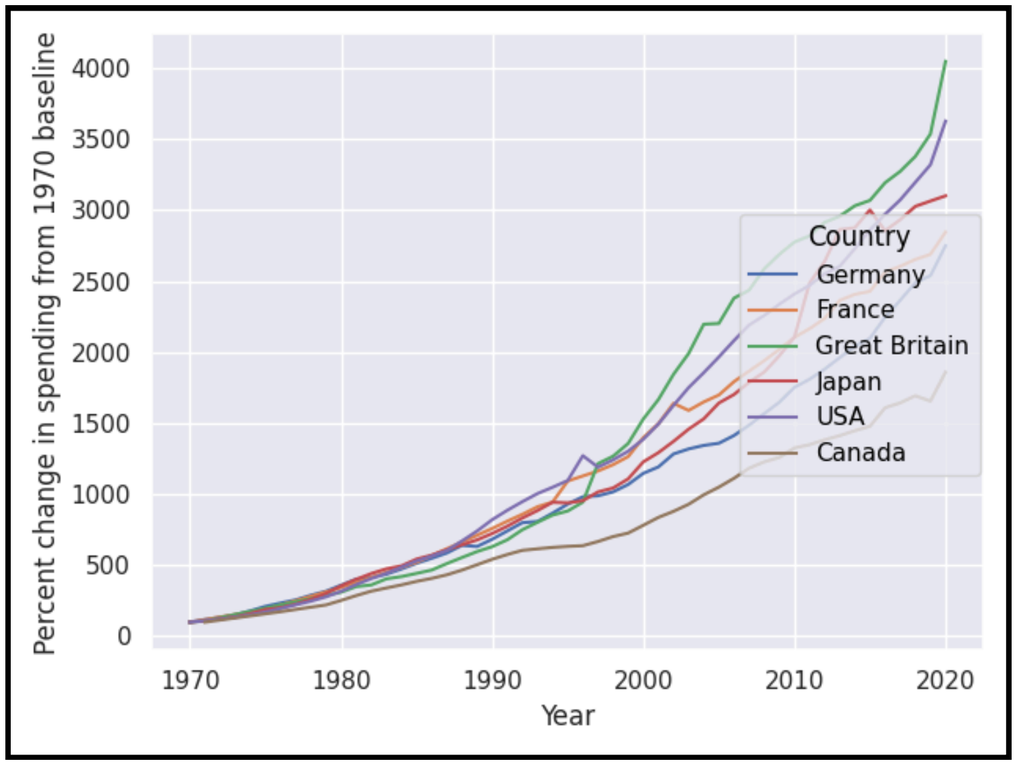
L'objet Perc() permet d'afficher les centiles de données. Si l'on ne choisit pas nous avons les 20ième, 40ième, 60ième, 80ième et 100ième centiles. On peut donner une liste pour avoir précisément ce que l'on veut.



En combinant avec Range() et Shift() pour décaler l'affichage on obtient une ligne représentant l'étendue des centiles demandés.

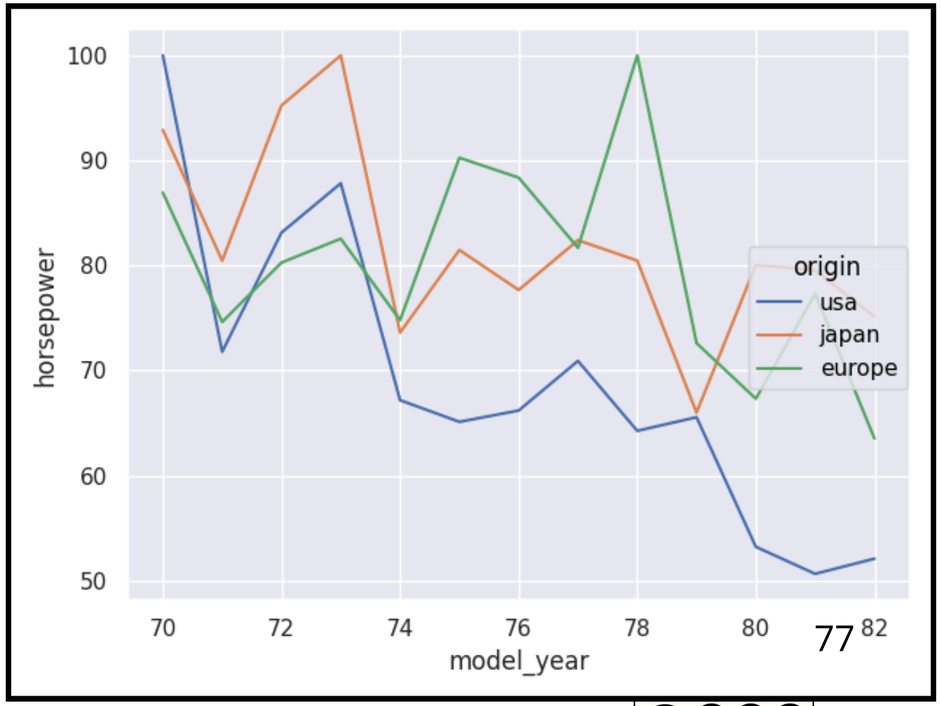
```
so.Plot(diamonds, "price", "cut").add(so.Dots(pointsize=1, alpha=.2),
so.Jitter(.3)).add(so.Range(color="k"), so.Perc([0, 25]),
so.Shift(y=.2)).add(so.Range(color="k"), so.Perc([75, 100]),
so.Shift(y=.2)).scale(x="log").show()
```

NORM




```
so.Plot(healthexp, x="Year", y="Spending_USD", color="Country").add(so.Lines(), so.Norm(where="x == x.min()",percent=True)).label(y="Percent change in spending from 1970 baseline").show()
```

L'objet Norm() permet de normaliser les données selon l'axe de valeurs. On peut utiliser where et une requête afin de choisir à partir de quoi on normalise. Ici on utilise la valeur de x minimum, c'est à dire l'année minimum. On a donc la variation en pourcentage au fil des années selon la valeur de l'année la plus ancienne.

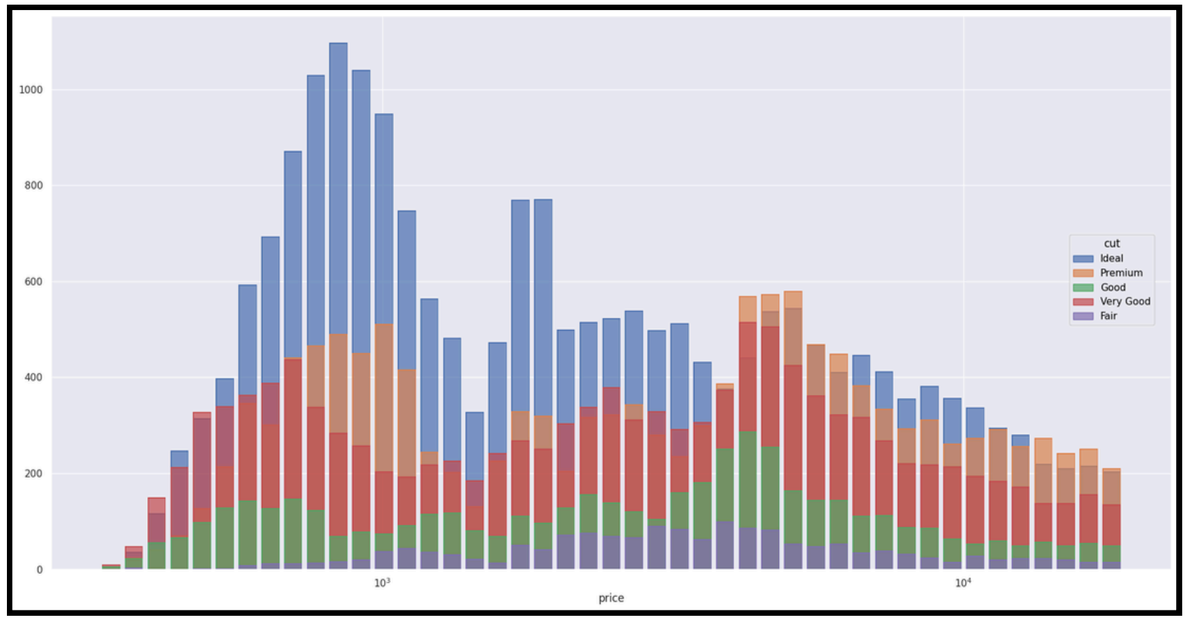


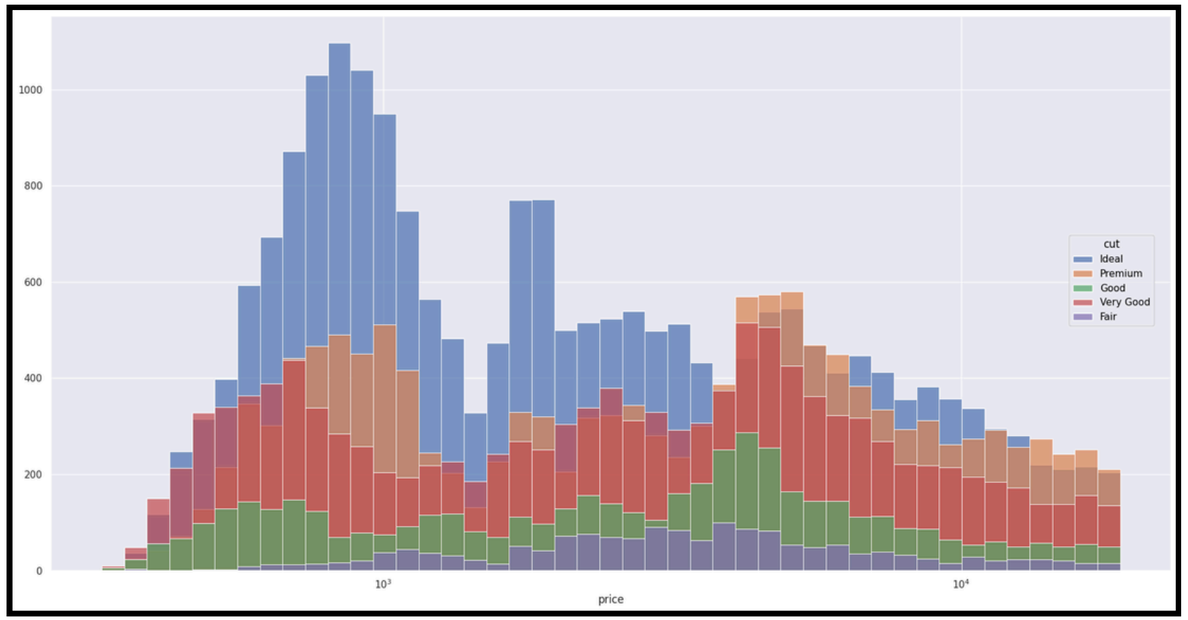

```
so.Plot(mpg,"model_year","horsepower").add(so.Line(),so.Agg(),so.Norm(percent=True),color="origin").show()
```

Autre exemple montrant l'évolution de la puissance de différents modèles de voitures au fil des ans pour les USA, l'Europe et le Japon, 100% étant la valeur maximale atteinte pour chaque région.

VARIANTES AVEC S

Dot(), Line(), Path() et Bar() ont tous des variantes finissant par un s. Le but de ces versions est de pouvoir afficher des graphiques qui soient lisibles même avec beaucoup de données. Par exemple Dots créer des cercles non remplis, dont on peut contrôler l'intérieur pour faire apparaître une autre variable. Toutes ces versions sont plus performantes pour tracer beaucoup de données sur un graphique ce qui peut devenir long avec les versions simples. Voici un exemple avec Bar à gauche et Bars à droite :





SCALES

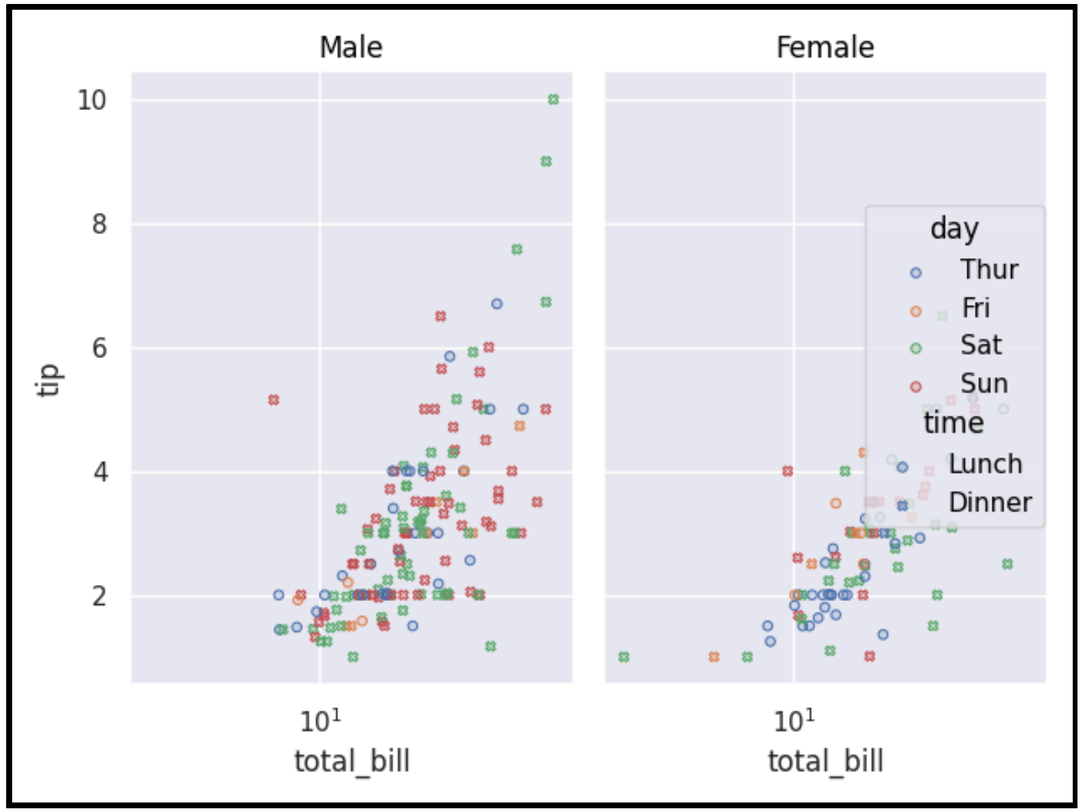
On peut aussi contrôler l'échelle des différents axes avec l'objet .scale(). Seaborn analysera automatiquement le type de données pour choisir si l'échelle des valeurs doit être numérique(Continuous()), catégorielle(Nominal()), une date(Temporal()) ou booléenne(Boolean()).

Au sein par exemple de Continuous on peut contrôler le type d'échelle numérique que l'on utilise

```
so.Plot(tips,x="total_bill",y="tip").add(so.Dots(),so.Jitter(0.5),color="day",marker="time").facet("sex").scale(x=so.Continuous(trans="log"),y=so.Continuous(trans="sqrt")).show()
```

- L'échelle logarithmique de base 2 est disponible avec log
- Celle de base 10 avec log10
- Racine carrée avec sqrt

C'est possible de forcer une échelle catégorique sur des données numériques mais si ces données ne sont pas une petite liste d'entier ca sera vite illisible.
Si l'on utilise une échelle booléenne sur des données numériques le graphe sera visible du côté True et le côté False sera vide.



ISDM

INSTITUT DE SCIENCE DES DONNÉES MONTPELLIER

UNIVERSITÉ DE MONTPELLIER

Inserm

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

Montpellier Méditerranée Métropole

Contrats de plan ÉTAT-RÉGION

FRANCE 2030

anr agence nationale de la recherche

L'EUROPE S'ENGAGE L'OCCITANIE AGIT

La Région Occitanie Pyrénées - Méditerranée

+33 (0)4 67 14 47 89

isdm.umontpellier.fr

Université de Montpellier
Bât. 4 et 15 – CC 13004
Place Eugène Bataillon
34095 Montpellier Cedex 5

MERCI